UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Título: AstroCódigo. Un juego serio para la introducción de jóvenes en los conceptos básicos de la programación.
Autores: Bione Javier, Miceli Pablo
Director: Sanz Cecilia
Codirector:
Asesor profesional: Artola Verónica
Carrera: Licenciatura en Informática, Licenciatura en Sistemas

Resumen

La presente tesina se enmarca en un proceso de investigación aplicada relacionada con la inclusión de tecnologías digitales, en especial el uso de juegos, en procesos de enseñar y aprender. Se realizó un estudio bibliográfico sobre juegos serios, y como resultado se ha problematizado su definición en la introducción de este trabajo. A la vez que se procedió a la revisión de antecedentes de juegos serios para la enseñanza de la programación junto con un análisis de las posibilidades ofrecidas por dichos juegos. Luego, se definieron los requerimientos del juego serio a diseñar en función de los aspectos de éxito de otros juegos relacionados al objetivo del propuesto aquí, y también en base a los aspectos aún no contemplados en estos antecedentes revisados. Seguidamente, se realizó el diseño y la implementación del juego serio AstroCódigo, a partir de los requerimientos determinados. Se realizaron sesiones de prueba del juego con alumnos en un escenario educativo concreto, considerando como ejes de evaluación: la usabilidad, posibilidades para la adquisición de habilidades relacionadas a la programación y la motivación intrínseca durante su utilización. Por último, se procedió al análisis de los resultados obtenidos en las sesiones y a la planificación de una propuesta de mejoras.

Palabras Claves

Juegos Serios, Programación, Gamificación, Objetivos Educativos, Videojuegos, Aprendizaje, Algoritmos, Estructuras de Control, Variables, Módulos.

Trabajos Realizados

- Revisión bibliográfica sobre definición y antecedentes de juegos serios para la enseñanza de la programación junto con un análisis de las posibilidades ofrecidas por estos juegos.
- Definición de los requerimientos del juego serio a diseñar en función de los aspectos de éxito de otros juegos relacionados al objetivo del propuesto aquí.
- Diseño, implementación y evaluación del prototipo de juego serio AstroCódigo a partir de los requerimientos determinados. Análisis de resultados.
- Creación de una herramienta de generación de nuevos escenarios de programación para AstroCódigo, con el fin de que los docentes puedan personalizar el juego.

Conclusiones

Mediante las sesiones de prueba realizadas en el marco de esta tesina se pudo observar que AstroCódigo ofrece un aporte a la comunidad, siendo éste un juego orientado tanto para la enseñanza como para el aprendizaje de conceptos relacionados a la programación. Se ha visto que permite identificar intuitivamente el concepto de estructura de control y su necesidad en la creación de algoritmos. Los estudiantes remarcaron haber profundizado en la comprensión de las estructuras de control abordadas y en la creación de algoritmos.

Trabajos Futuros

Se proponen mejoras sobre el prototipo desarrollado tales como: incluir de manera práctica los conceptos de variables y módulos, mejorar el sitio web del juego y perfeccionar el algoritmo de detección de patrones de instrucciones. Por otro lado, se propone: realizar nuevas evaluaciones con mayor cantidad de participantes para obtener resultados más concluyentes, realizar sesiones de pruebas con docentes, y ampliar la base teórica con investigaciones hechas posteriormente a la indagación presentada en este documento.

Fecha de la presentación: Junio de 2017

Índice general

CAPÍTULO 1

Introducción

1.1. Introducción

En este capítulo se presentan los objetivos propuestos para la tesina de grado, la motivación, la metodología seguida para llevar adelante el trabajo, los resultados esperados y la estructura general del informe. De esta manera, se detallan los fundamentos del trabajo y se prepara al lector con una mirada general para luego recorrer cada capítulo. En la siguiente sección se presenta la motivación del trabajo.

1.2. Motivación

En la actualidad las tecnologías informáticas desempeñan un papel fundamental en muchos aspectos de la vida cotidiana de las personas, como en la educación, el ámbito laboral, y hasta en los momentos de ocio. Sin embargo, el explosivo incremento de la popularidad de las tecnologías informáticas no se ve reflejado de igual forma en la educación escolar, donde no es raro que se encuentren jóvenes que ignoren por completo las nociones y conceptos básicos sobre el funcionamiento y programación de computadoras. Incluso alumnos ingresantes de carreras orientadas a las Tecnologías de Información y Comunicación

(TICs), muchas veces presentan dificultades sobre estos mismos conceptos básicos en lo que refiere a programación (Queiruga, Fava, Gómez, Miyuki Kimura & Brown Bartneche, 2014; López, Del Olmo, Reyes & Fernández, 2013).

La problemática anteriormente mencionada repercute directamente en la industria informática, ya que ésta demanda una cantidad de profesionales muy superior a la que genera el sistema educativo argentino. Según el Observatorio Permanente de la Industria de Software y Servicios Informáticos de la Argentina (OPSSI), en su reporte correspondiente al año 2014, en los últimos diez años se incrementó en un 132 % el número de empresas destinadas a los servicios informáticos. Paralelamente, según el Análisis de Diagnóstico Tecnológico Sectorial (Software y servicios informáticos) del Ministerio de Ciencia, Tecnología e innovación Productiva, más de la mitad de las empresas manifestaron tener una alta dificultad para cubrir su demanda de: líder de proyecto (54 %), desarrolladores web (cerca de 70 %), analista de sistemas funcional (alrededor de 65 %), desarrollador cliente-servidor (60 %), arquitecto de soluciones (76 %), administrador de base de datos (55 %), director de proyectos (69 %), administrador de seguridad (60 %) y especialista en seguridad de aplicaciones (70 %) (Cámara de Empresas de Software & Servicio Informáticos de La República Argentina, 2014; Ministerio de Ciencia, Tecnología e Innovación Productiva, 2014).

Este fenómeno crea una brecha entre la tecnología y la gente, provocando que a medida que la tecnología cobra más relevancia y se entrelaza cada vez más en diversos ámbitos de la vida cotidiana, no haya la suficiente cantidad de profesionales capacitados para suplir la demanda de trabajo en las áreas de Informática. Se cree que uno de los principales motivos de esta situación es el hecho de que los jóvenes no se ven expuestos a conceptos o experiencias de programación adecuadas en edades tempranas, dificultando así que estos se interesen por las carreras afines a la informática.

La Informática y los conceptos circundantes a la programación de computadoras pueden ser percibidos a simple vista como tediosos, aburridos e incluso algo complejos por un joven en edad escolar. Esto llevado a varios investigadores y educadores a pensar que para,

realmente, lograr un interés de los jóvenes por la Informática y la programación, se deben abordar estos conceptos por medio de estrategias alternativas o no-tradicionales como por ejemplo, los videojuegos.

Para los jóvenes de hoy en día, los videojuegos son algo muy común. Algo con lo que están acostumbrados a interactuar pues les generan interés y permiten visualizar conceptos, ideas y procedimientos de una manera que les es más amena y familiar a su propia cultura. Les permite asumir un rol donde se enfrentan a desafíos, aprenden de ellos y de sus acciones en el juego, al mismo tiempo que interactúan con otros jóvenes que les ayudan. Es por esto que la utilización de los juegos serios es una manera de introducir los conceptos que puede resultar más atrapante e interesante para ellos (Díaz, Queiruga & Fava, 2015; Gomes, Pereira Lopes & Vaz de Carvalho, 2013).

1.3. Objetivos

1.3.1. Objetivo general

El objetivo, a nivel general, es estudiar la utilización de juegos serios para la introducción de conceptos básicos sobre la programación de computadoras en jóvenes de distintas edades.

1.3.2. Objetivos específicos

Analizar estrategias de enseñanza, factibles y adecuadas, para ser consideradas en el desarrollo de juegos serios educativos, poniendo especial énfasis en aquellos orientados a la enseñanza de la programación.

Investigar sobre casos de éxito sobre la utilización de juegos serios educativos para introducir nuevos conceptos o desarrollar habilidades en jóvenes. En esta investigación se considerarán aspectos educativos y tecnológicos de cada experiencia revisada.

Diseñar y desarrollar un juego serio con la consideración de la investigación previamente realizada.

Realizar una prueba de campo del juego desarrollado, con el fin de validar las posibilidades y barreras encontradas en una muestra de jóvenes estudiantes e ingresantes de distintas carreras.

1.4. Metodología de trabajo

Para cumplir los objetivos propuestos, primero se procederá a definir el concepto de juego serio y su importancia en el ámbito educativo. Se revisarán antecedentes de juegos serios, poniendo especial énfasis en las experiencias que involucran aprendizajes orientados a habilidades de programación, y destacando aquellas que hayan demostrado su efectividad.

Luego, se estudiará la Taxonomía de Bloom, herramienta que se utilizará para la clasificación de los objetivos educativos del juego a desarrollar. También se estudiarán las operaciones del pensamiento de orden superior para profundizar en los procesos cognitivos que ayuden en el aprendizaje del jugador.

Seguidamente, se diseñará el juego serio orientado al aprendizaje de conceptos relacionados a la programación, teniendo en cuenta tanto los casos de éxito relevados previamente sobre el uso juegos similares, como las operaciones del pensamiento de orden superior que el jugador tendrá que poner en práctica. Una vez concluida la etapa de diseño del juego, se comenzará con el desarrollo del mismo.

Por último, se llevará a cabo una prueba del juego desarrollado con una muestra representativa del grupo destinatario con el fin de analizar su eficacia en cuanto a los objetivos educacionales planteados y aspectos de usabilidad.

1.5. Resultados esperados

El principal resultado esperado de esta tesina, es el diseño y desarrollo de un juego serio que permita introducir a los jóvenes conceptos relacionados a la programación y despierte su interés en los mismos, basándose en casos de éxito en la utilización de juegos serios como referencia y en el estudio de los procesos cognitivos involucrados en el aprendizaje de estos temas. Al mismo tiempo, se deja como un aporte la indagación teórica relacionada con el tema que puede servir a otros interesados en la temática, el proceso realizado con un análisis de los desafíos encontrados desde el punto de vista del diseño y la implementación de un juego serio con un motor específico de desarrollo. Esto último se considera valioso ya que permitirá a otros desarrolladores tener en cuenta aspectos que se han logrado solucionar a lo largo del camino.

1.6. Estructura de la tesina

A continuación se detalla la estructura de esta tesina de aquí en adelante.

Capítulo 2: Juegos serios. En este capítulo se abordará el concepto de juego serio y su importancia en el ámbito educativo. Se revisarán antecedentes de juegos serios, poniendo especial énfasis en las experiencias que involucran aprendizajes orientados a habilidades de programación. Se finalizará el capítulo con un análisis de importantes casos de éxito de la utilización de juegos serios en la educación informática.

Capítulo 3: Objetivos educativos. Se analizará la Taxonomía de Bloom y su posterior revisión a manos de Krathwohl como herramienta para la clasificación de objetivos educacionales. El capítulo concluirá con el análisis de las operaciones del pensamiento de orden superior que se verán involucradas en el juego a desarrollar para cumplir sus objetivos educativos.

Capítulo 4: Diseño del juego. Se detallarán los aspectos de diseño del juego a desarrollar.

Se comenzará comentando la idea general del mismo, para continuar con la especificación de los objetivos educacionales que perseguirá. Luego, se especificarán aspectos importantes del juego, como lo son la historia, las mecánicas de juego, los niveles y las decisiones generales tomadas.

Capítulo 5: Desarrollo. Se describirán las tecnologías utilizadas en el desarrollo del juego y las razones por las cuales fueron seleccionadas. También se mencionarán las dificultades encontradas y los cambios de rumbo en cuanto a su desarrollo, de manera tal de generar camino para otros desarrolladores a partir de las decisiones que se fueron tomando.

Capítulo 6: Prueba para la evaluación del juego. En este capítulo se detallará la metodología utilizada para realizar una evaluación del juego desarrollado. Se describirán las pruebas realizadas y se presentará un análisis de los resultados obtenidos.

Capítulo 7: Conclusiones y trabajos futuros. Para finalizar este trabajo, se explicarán las conclusiones alcanzadas y las posibles mejoras o extensiones que se pueden realizar sobre el juego.

CAPÍTULO 2

Juegos serios

2.1. Introducción

Como ya se mencionó anteriormente, uno de los principales objetivos de este trabajo es el de desarrollar un juego educativo orientado al aprendizaje de conceptos relacionados a la programación. En este capítulo se pone el foco en los juegos y sus oportunidades para el escenario educativo. Primero se describe el juego mismo como actividad propia del ser humano y se resalta su potencial como actividad educativa. Luego se aborda el concepto de juego serio, sus características y clasificaciones. Por último, se mencionan las ventajas del uso de juegos serios en el escenario educativo y se enumeran casos de éxito de juegos relacionados a la enseñanza de conceptos de programación, campo al cual se orienta el desarrollo de esta tesina.

2.2. El juego como actividad

Johan Huizinga (1938), en su libro *Homo Ludens*, define la actividad de jugar como actividades voluntarias donde se asume un rol y se pretende ser algo o alguien que no se es, creando escenarios imaginarios que pueden o no tener relación con la realidad, y

absorbiendo la completa atención de los jugadores. Los juegos se realizan en un tiempo y espacio específico, y son jugados en base a reglas establecidas creando grupos sociales entre sus jugadores.

Jugar tiene un alto potencial como actividad educativa, ya que los jugadores pueden interpretar subjetivamente los procedimientos multimodales que crea un juego mientras que estos se relacionan con el conocimiento del *mundo real* preexistente en el jugador. Además, los juegos brindan la posibilidad de experimentar de forma segura sobre situaciones que de otra forma no podrían ser experimentadas en el *mundo real*. Teóricos del juego contemporáneo aseguran que, a través de los juegos se pueden contar historias con efectos reales en los jugadores, desarrollar y explorar identidades a través de *avatares*, leer retórica cultural, y educar. El aprendizaje puede ocurrir a través de la experiencia, la interacción, el diseño del juego y el compromiso con el juego, siendo este aprendizaje incluso más contextualizado que la estructura descontextualizada basada en hechos de la educación tradicional (Seel, 2012).

Puede que parezca que el uso de juegos en la educación sea algo reciente, sin embargo, los juegos son utilizados prácticamente desde siempre. Por ejemplo, hace alrededor de mil años, los oficiales militares utilizaban juegos como el Ajedrez, para enseñar conceptos importantes relacionados a tácticas y estrategias de guerra. Sin ir más lejos, y tal y como lo recomendó Platón en su obra *La República*, los juegos son utilizados en la educación de los niños en etapas tempranas. Concretamente en preescolar se utilizan juegos como actividades socializadoras y para preparar a los niños para un aprendizaje ordenado (Chen & Michael, 2006).

2.3. El concepto de juego serio

Usualmente, cuando una persona se dispone a jugar, lo hace con el fin de obtener diversión y entretenimiento de ello. Sin embargo, puede que el jugador se tope, incluso sin darse cuenta, con distintos tipos de aprendizajes durante la actividad. Cuando un juego es

creado con el principal objetivo de que el jugador obtenga estos aprendizajes, se dice que el juego tiene *objetivos caracterizantes*, los cuales apuntan a desarrollar en los jugadores competencias que pueden categorizarse en seis dominios: cognitivas y perceptivas, tales como percepción, atención, entendimiento, pensamiento estratégico, resolución de problemas, administración y planificación, y memoria; emocional y volitiva, tales como control emocional, control de estrés y resistencia; personales, tales como auto-observación, autocrítica, autoeficacia, identidad y control emocional; sensoriales y motoras, tales como coordinación, tiempo de reacción, balance, ritmo, orientación, flexibilidad, resistencia y fuerza; sociales, tales como cooperación, competitividad, soporte mutuo, empatía, interacción, comunicación y juicios morales; y de medios de comunicación, tales como conocimiento de medios de comunicación, uso autorregulado, comunicación activa y diseño de medios de comunicación. Sin ir más lejos, cualquier juego cuyo objetivo primario sea desarrollar alguna de estas competencias o habilidades en el jugador, puede ser llamado *juego serio* (Dörner, Effelsberg, Göbel & Wiemeyer, 2016).

Hasta el día de hoy, no existe una definición universal de *juego serio* aceptada. Incluso en algunas definiciones los *juegos serios* no están caracterizados por la intención del desarrollador a la hora de crearlo, sino en la del jugador a la hora de jugarlo, por ejemplo, una persona podría elegir jugar un juego de rugby con el principal propósito de aprender las reglas del deporte (Dörner, Effelsberg, Göbel & Wiemeyer, 2016). Otra definición describe a los *juegos serios* como aquellos que no tienen al entretenimiento, el disfrute o la diversión como su propósito primario, lo cual no quiere decir que los *juegos serios* no sean entretenidos y divertidos, solo que fueron creados con otros motivos (Chen & Michael, 2006). Mientras algunas definiciones sostienen que un *juego serio* debe ser a la vez un *juego digital* o *video juego*, es decir, un juego que utilice algún tipo de máquina de cómputo, otras se refieren a todo tipo de juegos en general, de hecho, Clark Abt, formalizó el término *juego serio* teniendo en mente juegos de mesa y de cartas (Abt, 1970). En esta tesina se adoptará la definición que afirma que un *juego serio* es un juego digital creado con al menos un *objetivo caracterizante*, además del de entretener (Dörner, Effelsberg, Göbel & Wiemeyer, 2016).

Los juegos serios están enmarcados en el concepto de *entretenimiento educativo* o *edutainment* el cual refiere al género híbrido de combinar aprendizaje y diversión. La idea de *edutainment* es la de captar y mantener la atención de los estudiantes abordando sus emociones a través de contenido visual, narrativo y parecido a juegos (Seel, 2012). A su vez, cabe aclarar la diferencia entre *juegos serios* y *gamificación* o *ludificación*, siendo este último un concepto que refiere a la utilización de elementos y dinámicas propias de los juegos, en actividades no relacionadas a juegos con el fin de potenciar la motivación de aquellos que las realizan. Los juegos serios pueden clasificarse acorde a diferentes criterios. Algunos de estos criterios son: el dominio de las competencias dentro de sus *objetivos categorizantes*, el público objetivo, el área de aplicación, o el nivel de formalidad de su contenido, ya sea éste contenido educativo, simulación o entrenamiento. Sin embargo, de acuerdo con el *Sistema de Clasificación de Juegos Serios* creado por *Ludoscience*[1] y por el *Directorio de Juegos Serios* provisto por la *Asociación de Juegos Serios*, las categorías de juegos serios más populares son: *Juegos corporativos para entrenamiento y simulación*, *juegos educativos*, *juegos de salud*, *juegos publicitarios*, *juegos de conciencia social*, *juegos de arquitectura y planificación*, *juegos de turismo* y *juegos de herencia cultural*. De todas estas categorías cabe destacar la de *juegos educativos*, que refiere a los *juegos serios* cuyo objetivo principal se orienta al logro de un aprendizaje por parte de quienes lo juegan, y a la cual pertenece el juego que se diseña en esta tesina (Dörner, Effelsberg, Göbel & Wiemeyer, 2016).

2.4. Los juegos serios en escenarios de aprendizaje

Ya se han mencionado las posibilidades de los juegos para entrenar, aprender y desarrollar distintas competencias y habilidades. A continuación se exploran las potencialidades de los juegos serios en escenarios de aprendizaje. Los juegos serios pueden ser utilizados desde el jardín hasta en la educación superior, y en diferentes campos de la educación formal e informal. En diversas disciplinas posibilitan a las personas recrear situaciones que

[1]Laboratorio de investigación y desarrollo orientado a múltiples formas de juego.

son difíciles de realizar en la vida real, o que resultan riesgosas. Cada vez aparecen más juegos serios asociados a abordar formas de comunicación, negociación y trabajo en equipo (Rugelj, 2015).

También destacan su importancia en el *aprendizaje experimental* que enfatiza un proceso de construcción de conocimiento activo por medio de transacciones entre la persona y el entorno. En este proceso la persona utiliza sus conocimientos previos y formas de pensar en la construcción del conocimiento, dando como resultado la adquisición o desarrollo de varias habilidades tales como autoreflexión, resolución de problemas y pensamiento abstracto, al mismo tiempo que aprenden a controlar sus emociones, probar sus límites y a desarrollar habilidades complejas de comunicación. Por otro lado, uno de los argumentos más articulados a la hora de utilizar juegos serios, se vincula con la *motivación intrínseca*, la cual es una función de una relación óptima entre desafío, fantasía, curiosidad y control. Se afirma que los estudiantes al estar altamente motivados a través de experiencias de aprendizaje ricas, interesantes y atractivas, mejoran su entendimiento. Por otro lado, los estudiantes tienden a prestar más atención a información que es presentada de manera dinámica. Además, se argumenta que cuando los estudiantes se divierten, están más motivados a permanecer más tiempo en una actividad o a abordar la actividad más seguido (Seel, 2012). Todos estos argumentos y otros tantos que abordan diversidad de autores en la temática, despiertan la iniciativa para continuar profundizando en las investigaciones y el avance en la integración de estos juegos en escenarios de aprendizaje. A continuación se hará un recorrido por una serie de juegos serios vinculados a la enseñanza y aprendizaje de la programación.

2.5. Juegos serios en la enseñanza y aprendizaje de la programación

A continuación se describen antecedentes en la utilización de juegos serios para la introducción de conceptos de programación en jóvenes y otras experiencias con uso de diferentes herramientas para la enseñanza de estos temas, con el fin de recuperar características de

interés para el juego que se diseña y desarrolla como parte de este trabajo. En concreto se analizan ocho casos de éxito los cuales sirven como punto de partida para la especificación de los requerimientos del juego.

2.5.1. *Program your robot*

Program your robot es un juego serio diseñado para que estudiantes practiquen conceptos introductorios a la programación, tales como construcción de algoritmos, *debugging* y simulación. El objetivo del juego es construir una solución algorítmica que permita ayudar a escapar a un robot de distintos escenarios. El estudiante tiene a su disposición bucles de iteración, condicionales e instrucciones simples, pudiendo también crear funciones con todas éstas para su posterior reutilización. El algoritmo se construye arrastrando y soltando bloques desde un menú hacia el programa. El juego cuenta con seis niveles de complejidad incremental. En cada uno de estos niveles se encuentran dispersos, de forma aleatoria, objetos que el robot debe agarrar antes de escapar del escenario.

Se realizó una prueba de *Program your robot* con estudiantes de distintas carreras afines a las Ciencias de la Computación de la Universidad de Greenwich, entre los cuales, variaba considerablemente su conocimiento en programación. Los resultados de la prueba fueron positivos en cuanto a que los estudiantes percibieron el juego como una buena forma para ayudar a estudiantes a entender conceptos básicos de la programación, y a desarrollar habilidades en cuanto a la resolución de problemas (Bacon, Kazimoglu, Kiernan & Mackinnon, 2012).

2.5.2. *Software KIDS*

Software Kids es un juego serio desarrollado para dispositivos Android que permite introducir a niños a partir de los ocho años, principalmente, a conceptos básicos de la programación orientada a objetos, como por ejemplo: clases, objetos, atributos, métodos,

herencia, polimorfismo, abstracción, encapsulación, ocultamiento y modularidad. También se tuvieron en cuenta conceptos como algoritmos, estructuras condicionales e iterativas y arreglos. Además se incluyen las etapas comunes que componen la forma de resolver problemas de programación, tales como análisis de problemas, especificación de requerimientos, diseño, implementación, *testing*, documentación y mantenimiento. La mecánica del juego se basa en contestar preguntas o resolver pequeños enunciados ya sea tocando la respuesta correcta u ordenando elementos en la pantalla. Dichas preguntas y enunciados hacen alusión a conceptos de programación utilizando elementos de la vida cotidiana.

El juego fue probado con doce chicos de distintas edades, promediando una edad de entre nueve y diez años. La prueba consistió en una clase preliminar de aproximadamente una hora de duración, en la cual se introdujeron los conceptos que luego se abordan durante el juego. Luego de la clase, los niños procedieron a jugar *Software KIDS* durante cincuenta minutos, para luego finalizar la sesión con un cuestionario de diez minutos. Los resultados obtenidos arrojaron que en varios casos los niños indicaron que el juego resultaba de cierta dificultad para ellos, y que el lenguaje utilizado en algunos de los niveles era desconocido, sin embargo, se destacó la utilidad del juego tanto para introducir como para presentar conceptos relacionados a la programación orientada a objetos en niños de todas las edades (León-Sigg, Ramírez-Rosales, Vásquez-Reyes & Villa-Cisneros, 2016).

2.5.3. *Prog&Play*

El juego serio *Prog&Play* está basado en un juego de estrategia en tiempo real llamado *Kernel Panic*, el cual utiliza metáforas de las Ciencias de la Computación, tales como bits, punteros, etc. como unidades de juego. *Prog&Play* agrega a *Kernel Panic* una capa que permite a los estudiantes programar la inteligencia artificial de cada unidad del juego para ganarlo. La mecánica de juego se basa en que los estudiantes seleccionan una facción (Hackers, Sistemas y Redes) y hacen uso de las unidades que ésta les provee para derrotar a las demás haciéndolas combatir entre ellas.

La evaluación de *Prog&Play* se llevó a cabo en tres iteraciones. La primer iteración consistió en observar el comportamiento de los estudiantes mientras jugaban. Para la prueba se seleccionaron quince estudiantes de primer año de Ciencias de la Computación del Instituto de Tecnología de Toulouse, de entre los cuales ninguno tenía conocimientos en algún lenguaje de programación. Durante esta primera sesión los estudiantes jugaron *Kernel Panic* para familiarizarse con el juego, y luego se les introdujo *Prog&Play* para comenzar a utilizar programación para jugar. Los resultados de la primera iteración fueron satisfactorios, ya que los estudiantes demostraron interés en el juego. La segunda iteración de la evaluación se llevó a cabo en una escala más grande, integrando el juego en una clase con más de trescientos estudiantes. La prueba se realizó dividiendo la clase en dos, un grupo trabajando en la manera tradicional (grupo control) y el otro grupo que trabajó con *Prog&Play*. Los estudiantes que utilizaron *Prog&Play* demostraron menores fallos en las evaluaciones que aquellos pertenecientes al grupo de control, por lo que se puede decir que los resultados fueron positivos. La tercera iteración se basó en poner el juego en una página web para que terceros puedan incluirlo en sus propios contextos educativos. En esta última iteración, se llevaron a cabo dos experimentaciones con terceros, una de ellas en el Instituto de Tecnología de Toulouse y otra en la Universidad de Pierre y Marie Curie. En esta última iteración, los profesores a cargo de las experimentaciones demostraron interés en seguir utilizando *Prog&Play* en sus propuestas educativas (Jessel, Mauratet, Torguet & Viallet, 2011).

2.5.4. *Scratch*

Scratch no es un juego serio en sí, sino más bien un entorno de programación que permite la fácil creación de historias interactivas, animaciones, juegos, música y arte. Está diseñado para ayudar a niños a partir de los ocho años de edad a desarrollar habilidades de programación a medida que crean proyectos y se enfrentan a temas matemáticos y computacionales. El entorno cuenta con una interfaz muy amigable que permite crear *scripts* arrastrando y soltando bloques. Los bloques disponibles están separados por categorías

tales como, movimiento, la cual contiene bloques para moverse hacia delante, doblar, etc.; apariencia, cuyos bloques se encargan de controlar los disfraces del personaje y los fondos del escenario; sonido; lápiz; control, encontrándose entre sus bloques las estructuras de control más comunes; números, con bloques para operaciones aritméticas; sensores, cuyos bloques sirven para controlar los distintos sensores tales como el contacto entre *sprites*; y por último variables, cuyos bloques permiten almacenar, crear, imprimir valores, etc. Todos estos bloques dan una gran versatilidad al entorno, permitiendo crear proyectos de lo más variados (Kereki, 2008).

Scratch es un entorno de programación bastante popular y consolidado que ha sido puesto a prueba en numerosas ocasiones, a continuación se destacan tres de ellas. Darhmaoui et al. (2014) llevaron a cabo una prueba de *Scratch* con tres grupos de estudiantes de una secundaria con orientación científica en la ciudad de Nador, Marruecos. Uno de los grupos trabajó con *Scratch* para aprender conceptos de programación, mientras que los otros dos grupos sirvieron como grupos de control y se les enseñaron los mismos conceptos de forma tradicional. Los resultados de las encuestas posteriores determinaron que un alto porcentaje de los alumnos que había trabajado con *Scratch* disfrutó de la actividad, que incluso lo habían instalado en sus casas y que consideran seguir estudiando la materia en el futuro. Mientras que un gran procentaje del otro grupo manifestó que programar en entornos tradicionales le pareció aburrido y no se mostraron tan motivados para continuar estudiando programación (Darhmaoui, Elachqar, Kaddari, Lahmine & Ouahbi, 2014).

Se obtuvieron resultados similares al incorporar *Scratch* en las primeras tres semanas de uno de los cursos para ingresantes de la Universidad ORT de Uruguay, donde, si bien los conocimientos no variaron entre los grupos de experimentación con *Scratch* y el grupo control, los primeros demostraron mucho más interés y entusiasmo en seguir aprendiendo programación (Kereki, 2008).

La Universidad de Washington Bothell obtuvo los mismos resultados al realizar actividades con *Scratch* con estudiantes de escuela media, logrando que una gran parte de ellos se inscriba a cursos de programación extracurriculares (Gruenbaum, 2014). Este antecedente

refuerza la importancia del rol protagónico de los alumnos en el aprendizaje de la programación a partir de la creación de proyectos, y las potencialidades de crear algoritmos a partir de arrastrar bloques que representan instrucciones. Este último aspecto será considerado en el diseño del juego que se propone en esta tesina.

2.5.5. *RITA*

RITA es un juego serio desarrollado en Java y basado en los frameworks de código abierto *OpenBlocks* y *Robocode*. Mediante la programación en bloques provista por *OpenBlocks*, *RITA* permite definir las estrategias de combate de los robots virtuales que compiten en el campo de batalla de *Robocode*. La gramática de *RITA* se basa en una colección de bloques gráficos. Los estudiantes construyen sus programas arrastrando y soltando bloques y encastrándolos entre sí. La sintaxis no contiene signos de puntuación, ni palabras en inglés, propias de los lenguajes de programación. Los prerrequisitos académicos para jugar *RITA* son mínimos, por lo que cualquier estudiante de nivel secundario puede jugarlo sin mayores dificultades.

RITA fue puesto a prueba con alumnos de las escuelas participantes del proyecto *JET* en la Facultad de Informática de la Universidad Nacional de La Plata, Argentina. Gracias a los resultados positivos obtenidos en las pruebas, se logró incorporar *RITA* como parte de la currícula del Taller de Programación de dichas escuelas (Brown Bartneche, Gómez, Fava, Miyuki Kimura & Queiruga, 2014).

2.5.6. *EPRA*

EPRA (Enseñando Programación con Realidad Aumentada) es un material educativo digital que utiliza Realidad Aumentada para el desarrollo de una serie de actividades con diferentes intenciones didácticas para la enseñanza de Programación. Este material educativo consiste en un sitio web a través del cual los alumnos pueden complementar los

conceptos teóricos y prácticos vistos en los cursos introductorios de Programación de las carreras de la Facultad de Informática de la Universidad Nacional de La Plata, Argentina. Esto se logra mediante el uso de un material educativo que incluye actividades de Realidad Aumentada que combina marcadores en la escena real y al mismo tiempo la detección del rostro de la persona que está frente a la cámara para poder incluir objetos sintéticos. Las actividades con RA incluyen elementos lúdicos. La actividad de integración es un juego en sí mismo.

EPRA fue puesto a prueba tanto con alumnos ingresantes de la carrera Ingeniería en Computación de la Facultad de Informática de la Universidad Nacional de La Plata, sino también con profesores y auxiliares. Los resultados arrojaron que la actividad presenta altos niveles de motivación y de interacción en cuanto al contenido y que la sensación general al realizarla fue satisfactoria. Además, este tipo de actividades constituye una forma innovadora de enseñar (Salazar Mesía, Gorga & Sanz, 2015). Este antecedente si bien no es un juego serio fue considerado por las características lúdicas que podrían ser consideradas para el diseño del juego que aquí se propone.

2.5.7. *EPIT*

EPIT (Enseñanza de la Programación usando Interacción Tangible) es una aplicación informática, con la dinámica de juego, en la que pueden intervenir un grupo de alumnos y que funciona sobre una *tabletop*, la cual constituye el espacio de interacción en conjunto con marcadores como forma de detección de los objetos físicos con los que se trabaja. *EPIT* busca ejercitar al alumno en el análisis de soluciones en forma colaborativa, propone el ejercicio de debatir y negociar cuál de entre varias soluciones es la correcta y por qué. Pone en juego la comprensión de los alumnos de los temas abordados en las asignaturas vinculadas a la programación, y habilidades de análisis y colaboración para que el grupo de alumnos pueda arribar a la solución del problema planteado (Artola, Gorga, Pesado & Sanz, 2014).

Se llevaron a cabo sesiones de prueba para evaluar la efectividad de *EPIT* arrojando resultados que sugerían fuertemente que los participantes presentaban altos niveles de motivación para con la actividad y alta disposición al trabajo colaborativo para lograr los objetivos del juego (Artola, Giacomantone & Sanz, 2013).

2.5.8. *Visual DaVinci*

Visual DaVinci es un lenguaje visual estructurado orientado a la enseñanza de programación. Se basa en un robot que puede realizar una serie de acciones sobre una ciudad que cuenta con 100 calles (horizontales) y 100 avenidas (verticales). En las intersecciones entre calles y avenidas (esquinas) el robot se puede encontrar con cierta cantidad de papeles y flores. El robot puede recolectar y depositar flores y papeles en la esquina en la que se encuentra parado. A su vez, las esquinas pueden presentar obstáculos que impedirán el paso del robot por allí, por lo cual se deberá encontrar un camino alternativo (Champredonde & De Giusti, 1997).

En versiones posteriores, se extendió la funcionalidad de este entorno de aprendizaje en dos trabajos de grado de la Universidad Nacional de Tierra del Fuego, Antártida e Islas del Atlántico Sur para incluir conceptos de programación concurrente y paralela (Aguil Mallea, Depetris, Feierherd, Pendenti, Prisching & Tejero, 2015). Su posterior prueba con alumnos de primer año de la Facultad de Informática de la Universidad Nacional de La Plata mostró un incremento en el interés por los conceptos relacionados a la concurrencia (Chichizola, De Giusti A., De Giusti L., Leibovich, Naiouf, Rodriguez & Sanchez, 2014).

2.6. Conclusiones del capítulo

En este capítulo se ha revisado la definición de juegos serios, y se ha analizado el uso de estos en escenarios educativos. A partir de la revisión realizada de bibliografía y de juegos serios orientados al aprendizaje de la programación, y otras herramientas

vinculadas al aprendizaje de esta temática se han encontrado algunas características comunes que parecieran resultar exitosas a la hora de desarrollar competencias vinculadas a la programación y se han encontrado otros aspectos no tenidos en cuenta en estos proyectos, que se creen podrían ser motivadores y prometedores para los jóvenes. A continuación se listan algunas de las características encontradas y que resultaron de interés para incorporar en el desarrollo del juego serio propuesto:

1. Creación de algoritmos a partir de arrastrar y soltar bloques que representan instrucciones. Esta característica ha sido utilizada en *Program your robot*, *Prog&Play* y *RITA*. También se ha utilizado en entornos de programación visual tales como *Scratch* que fue mencionado en el análisis realizado.

2. Utilización de niveles del juego que incorporan un nivel progresivo de complejidad en términos de la adquisición de competencias en el desarrollo de algoritmos, como en el caso de *Program your Robot*.

3. Compilación de los algoritmos y detección de errores como el que utiliza *Prog&Play*.

4. Incorporación de premios y recompensas que favorezcan la motivación de los alumnos como técnica de gamificación. Esto se ha visualizado en la mayor parte de los antecedentes revisados.

Adicionalmente, se identificaron algunas características que pueden ser mejoradas o incluidas en el juego presentado en esta tesina respecto a los casos estudiados, entre ellas:

1. Retroalimentación y asignación de categorías a los jugadores, de acuerdo a la calidad de la solución construida para resolver un escenario, contemplando la aplicación de las mejores prácticas en cuanto a la utilización de estructuras de control y la eficiencia de movimientos realizados. De esta forma se estaría motivando al estudiante no solo a resolver los distintos escenarios, sino que a hacerlo de la mejor forma posible.

2. Posibilidad de que los niveles o escenarios dentro del juego puedan ser personalizados para distintos contextos educativos.

3. Utilización de elementos gráficos y mecánicas de juego más inmersivas, tales como escenarios 3D, movimiento en primera persona y la inclusión de una narrativa.

4. Incorporación de instancias de decisión previas a la construcción de algoritmos, relacionadas con la elección de los recursos necesarios para la resolución de los distintos escenarios.

En el capítulo 4 se profundiza en estos requerimientos, y en una serie de requerimientos adicionales que permitirán ofrecer elementos de motivación y/o aprendizaje adicionales para incorporar en el juego.

CAPÍTULO 3

Objetivos educativos

3.1. Introducción

En este capítulo se introducen algunos conceptos que permiten dar contexto a los aportes de este trabajo. En particular, se describen diferentes taxonomías que permiten guiar la definición de objetivos educacionales, entre ellas se presenta la Taxonomía de Bloom y posteriores revisiones de ésta. Es de interés abordar esta temática debido a que luego se utiliza para la definición de los objetivos educativos del juego serio que se diseña en esta tesina.

3.2. La planificación de una experiencia educativa

Acorde con lo expresado por Ander Egg (1993), planificar es prever "hacer algo", pero lo que se quiere hacer no puede quedar en algo nebuloso, vago, inasible o difícilmente realizable. "Aquello" que se va a hacer tiene que expresarse en objetivos concretos y bien definidos, traducidos operacionalmente en metas de cara a obtener los resultados previstos. La planificación supone las posibilidades de una relación de causalidad entre lo decidido (programado), lo realizado (ejecutado) y los resultados obtenidos.

Al planificar una experiencia educativa, es necesario pensar en los objetivos, en cómo y cuándo enseñar para alcanzar esos objetivos, en la secuenciación de los contenidos que viabilizan el alcance de dichos objetivos, y en cómo se evaluará la experiencia educativa, entre otros componentes fundamentales de un proceso educativo. La definición de los objetivos en la planificación educativa resulta un tema sustantivo y el punto de partida esencial. Es por ello que en la próxima sección se abordará este aspecto, dado el carácter educativo con el que se orienta el juego serio que se diseña.

3.3. La importancia de los objetivos educacionales

Para diseñar una experiencia educativa es necesario definir sus objetivos educacionales de forma clara. La definición de los objetivos educacionales es el punto inicial para planificar, desarrollar, guiar y evaluar una actividad educativa. Los objetivos educacionales u objetivos de aprendizaje son enunciaciones que describen qué es lo que se espera que el estudiante sepa, entienda, o sea capaz de demostrar luego de completar el proceso de aprendizaje (Seel, 2012). Por ejemplo, se podría decir que el objetivo educativo del juego que aquí se diseña es claro: que los estudiantes aprendan conceptos de programación, y si bien esto es cierto a nivel general, es necesario identificar de forma precisa qué habilidades, conceptos, actitudes y/o valores se desea que el estudiante adquiera. Como se puede ver, el proceso de definir los objetivos educacionales de una experiencia educativa parece trivial a simple vista, sin embargo es una tarea que resulta compleja y no debe ser subestimada si se busca el éxito de una experiencia o actividad educativa (Anderson, 2005). A continuación se presenta la Taxonomía de Bloom que constituye una guía para la redacción de objetivos educativos.

3.4. Taxonomía de Bloom

En el año 1948, durante la primera convención anual llevada a cabo por la Asociación Psicológica de los Estados Unidos, surgió la idea de un sistema que permita una clasificación

de objetivos educacionales, que facilite la comunicación entre las personas encargadas de desarrollar las currículas educativas. De esta forma se brindaría un marco teórico que favorece el intercambio de recursos educativos entre profesionales, y la investigación sobre nuevos métodos de enseñanza. Durante la serie de convenciones que se realizaron, la comisión conformada por Benjamín Bloom, Max Engelhart, Edward Furst, Walker Hill y David Krathwohl estuvo a cargo del desarrollo de una taxonomía que clasifique los objetivos educacionales tanto en el plano cognoscitivo como en el afectivo (Bloom, Engelhart, Furst, Hill & Krathwohl, 1956). La intención original de la comisión encargada del desarrollo de la taxonomía era elaborar una clasificación completa en tres áreas: el área cognoscitiva, la cual incluye aquellos objetivos que se refieren a la memoria o evocación de los conocimientos y al desarrollo de habilidades y competencias técnicas de orden intelectual; la afectiva, que describe los cambios de interés, actitudes y valores, el desarrollo de apreciaciones y una adaptación adecuada; y la psicomotora, que incluye las competencias manipulativas y habilidades motoras (Bloom, Engelhart, Furst, Hill & Krathwohl, 1956). En la Figura 2.1 se muestra la estructura de la taxonomía de Bloom y sus categorías ordenadas en base a su complejidad, tal y como aparece en su publicación original en 1956. En la siguiente sección se avanza sobre una versión revisada (Anderson & Krathwohl, 2001) de la taxonomía y se explica brevemente para dar contexto a lo que se presentará en posteriores capítulos de este informe, dado que es la herramienta que se utiliza para la definición de los objetivos del juego que se diseña.

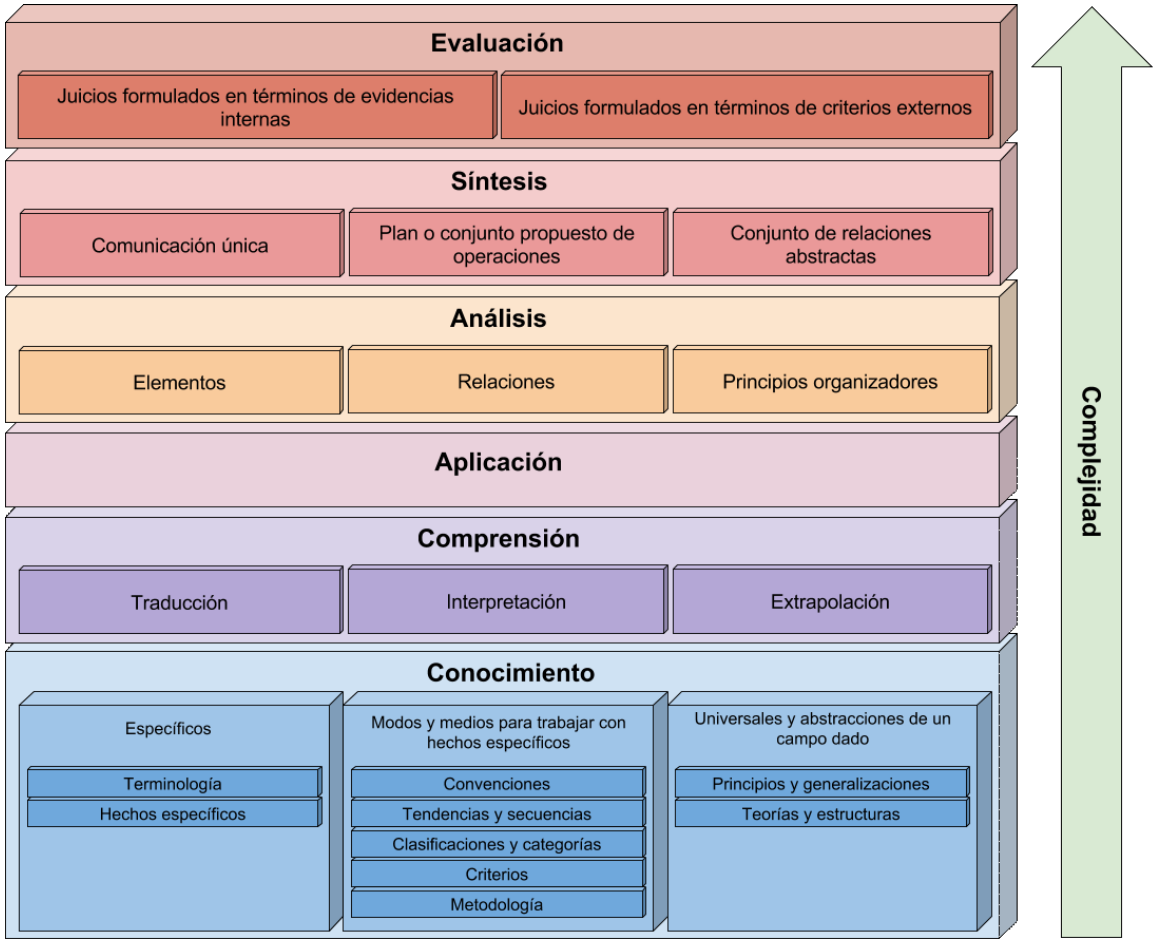

Figura 3.1: Estructura de la taxonomía de Bloom.

3.4.1. La taxonomía revisada por Krathwohl

Luego de 45 años de publicada la taxonomía de Bloom, David Krathwohl, uno de los contribuyentes originales de la misma, llevó a cabo una revisión que dio como resultado un herramienta que provee una forma de clasificar objetivos educacionales. Krathwohl detectó que a la hora de declarar los objetivos educacionales en forma de oración, éstas, naturalmente, se componían de un verbo o frase verbal y un sustantivo o frase sustantiva. Por ejemplo, si tomamos la siguiente frase: “El estudiante deberá ser capaz de recordar la ley económica de la oferta y demanda”, se puede identificar la frase sustantiva “ley económica de la oferta y demanda” y el verbo “recordar”. En esta revisión, la taxonomía pasa de una

dimensión a dos dimensiones, donde es posible enmarcar un objetivo educacional en la dimensión del *Conocimiento* en base al sustantivo de la oración, y en la dimensión del *Proceso Cognitivo* en base al verbo (Anderson & Krathwohl, 2001). A continuación se muestran de forma breve las dimensiones de conocimiento y de proceso cognitivo y sus respectivas estructuras.

Dimensión del Conocimiento

Esta dimensión es equivalente a la categoría de conocimiento de la taxonomía original, con algunos cambios en sus subcategorías. A continuación se definen las distintas categorías y subcategorías de esta dimensión (Anderson & Krathwohl, 2001).

A. *Conocimiento fáctico: elementos básicos que los estudiantes deben saber para familiarizarse con una disciplina o para resolver problemas dentro de ella. Esta categoría tiene a su vez dos subcategorías:*

    a) *Conocimiento de la terminología*

    b) *Conocimiento de elementos y detalles específicos*

B. *Conocimiento conceptual: las interrelaciones entre elementos básicos dentro de estructuras más grandes que les permiten funcionar de forma conjunta. Esta categoría tiene a su vez tres subcategorías:*

    a) *Conocimiento de categorías y clasificaciones*

    b) *Conocimiento de principios y generalizaciones*

    c) *Conocimiento de teorías, modelos y estructuras*

C. *Conocimiento procedural: cómo hacer algo; métodos de investigación y criterios para usar habilidades, algoritmos, técnicas y métodos. Esta categoría tiene a su vez tres subcategorías:*

    a) *Conocimiento de algoritmos y habilidades específicas del tema*

b) *Conocimiento de técnicas y métodos específicos del tema*

c) *Conocimiento de criterios para la determinación de procesos apropiados*

D. *Conocimiento metacognitivo: conocimiento de cognición en general así como conciencia y conocimiento de la cognición de uno mismo. La inclusión de esta categoría fue dada por su creciente importancia a medida que los investigadores demuestran la importancia de que los estudiantes estén conscientes de su actividad metacognitiva. Esta categoría tiene a su vez tres subcategorías:*

a) *Conocimiento estratégico*

b) *Conocimiento sobre tareas cognitivas, incluyendo conocimiento contextual y condicional apropiado*

c) *Autoconocimiento*

Dimensión del Proceso Cognitivo

La dimensión del Proceso Cognitivo contiene al resto de las categorías de la taxonomía original, con la diferencia que algunas de ellas sufrieron pequeñas modificaciones, renombres y cambios de orden (Anderson & Krathwohl, 2001). A continuación se definen cada una de las categorías de esta dimensión.

A. *Recordar: recuperar conocimiento relevante desde la memoria de largo plazo. Las subcategorías de este proceso cognitivo son:*

a) *Reconocer*

b) *Recordar*

B. *Entender: determinar el significado de mensajes instruccionales, incluyendo comunicación oral, escrita y gráfica. Las subcategorías de este proceso cognitivo son:*

a) *Interpretar*

b) *Ejemplificar*

c) *Clasificar*

d) *Resumir*

e) *Inferir*

f) *Comparar*

g) *Explicar*

C. *Aplicar: llevar a cabo o usar un procedimiento en una situación dada. Las subcategorías de este proceso cognitivo son:*

a) *Ejecutar*

b) *Implementar*

D. *Analizar: descomponer material en las partes que lo componen y detectar cómo dichas partes se relacionan las unas con las otras y cómo forman una estructura o propósito. Las subcategorías de este proceso cognitivo son:*

a) *Diferenciar*

b) *Organizar*

c) *Atribuir*

E. *Evaluar: Hacer juicios basados en criterios y estándares. Las subcategorías de este proceso cognitivo son:*

a) *Verificar*

b) *Criticar*

F. *Crear: poner elementos juntos para formar algo nuevo, un todo coherente o un producto original. Las subcategorías de este proceso cognitivo son:*

a) *Generar*

b) *Planificar*

c) *Producir*

Cómo utilizar la taxonomía

Esta nueva forma de clasificar los objetivos educacionales en base a dos dimensiones, presentada por la revisión de la taxonomía original, inmediatamente sugirió la utilización de una tabla bidimensional de clasificación tal como muestra la Tabla 2.1. El eje vertical de esta tabla representa la dimensión del conocimiento, mientras que el eje horizontal la del proceso cognitivo. De esta forma, cualquier objetivo educacional puede ser clasificado en una o varias intersecciones de ambas dimensiones (Anderson & Krathwohl, 2001).

	1.Recordar	2.Entender	3.Aplicar	4.Analizar	5.Evaluar	6.Crear
A.Conocimiento Fáctico						
B.Conocimiento Conceptual						
A.Conocimiento Procedural						
A.Conocimiento Metacognitivo						

Tabla 3.1: Tabla bidimensional de la taxonomía revisada de Bloom.

3.5. El cumplimiento de los objetivos

Tener los objetivos claros y clasificados apropiadamente es un gran avance, sin embargo, el verdadero desafío radica en crear una experiencia tal que se cumplan dichos objetivos. Para esto se deben tener presentes los pensamientos que se desean producir en los estudiantes a medida que experimentan las distintas situaciones presentadas en una experiencia educativa ¿Pero qué es pensar?

Cada vez que se resuelve un problema práctico, cuando uno se queda absorto y pensativo, cuando se ordenan los propios argumentos, cuando se demora en formular un juicio o tomar una decisión, cuando se analiza, se busca, se examina, y se investiga, lo que se hace es *pensar*. Se piensa *cómo*, se piensa *algo*, se piensa con *un fin*, se piensa *sobre algo* o se piensa *a fondo*. No es fácil dar una definición precisa y resumida sobre qué es el pensamiento de forma que

sugiera cabalmente toda la amplitud y profundidad de este concepto, sin embargo, Raths intenta resumirlo como un proceso asociado a la investigación y toma de decisiones (Raths & Wassermann, 1967).

Otra definición es la presentada por Wikipedia, en la que se define pensamiento como *actividad y creación de la mente, o todo aquello que es traído a existencia mediante la actividad del intelecto. El término es comúnmente utilizado como forma genérica que define todos los productos que la mente puede generar, incluyendo las actividades racionales del intelecto o las abstracciones de la imaginación; todo aquello que sea de naturaleza mental es considerado pensamiento, bien sean estos abstractos, racionales, creativos, artísticos, etc.*

En su libro *Cómo enseñar a pensar*, Raths (1967) propone una hipótesis indicando que las experiencias educativas que brindan la posibilidad de poner en práctica operaciones del pensamiento, suelen ser más enriquecedoras tanto en la transmisión de conocimiento sino también en la formación de individuos maduros. Entre dichas operaciones, se pueden identificar las siguientes como las relacionadas al aprendizaje: *Comparar; Resumir; Observar; Clasificar; Interpretar; Formular críticas; Búsqueda de suposiciones; Imaginar; Reunir y organizar datos; Formular hipótesis; Aplicar hechos y principios a nuevas situaciones; Toma de decisiones; Diseñar proyectos o hacer investigaciones; Codificar; Marcar con letras del "código" diferentes escritos.* Esta clasificación de operaciones del pensamiento, que aparecen como algunos de los verbos de las taxonomías presentadas, pueden resultar orientadoras también a la hora de planificar una experiencia o actividad educativa. Es por ello que se presentan aquí detalladamente la definición de algunas operaciones, según lo presentado por Raths. Las operaciones elegidas se relacionan con algunas que formarán parte de la experiencia educativa del juego que se diseña. En este capítulo sólo se explicarán las operaciones en sí acorde a lo presentado por Raths (Raths & Wassermann, 1967), dejando para el capítulo 4, donde se aborda el diseño del juego, la explicación sobre cómo estas operaciones deberán ser aplicadas para resolver cada situación presentada en el juego.

3.5.1. Comparar

Cuando una persona compara, emplea el pensamiento para observar diferencias y similitudes por la vía de los hechos y la contemplación. Examina dos o más objetos, ideas o procesos en búsqueda de puntos de coincidencia y no coincidencia. El proceso de comparar implica abstraer y retener mentalmente la abstracción, mientras concentra la atención en los objetos comparados. Si la comparación es practicada con una finalidad real y genuina, es decir, si hay una motivación verdadera en la búsqueda de lo semejante y lo desemejante, esta actividad termina siendo muy provechosa.

3.5.2. Observar

La observación encierra la idea de vigilar, reparar, notar y percibir. Usualmente se observa al tener un propósito definido, es decir, se concierne sobre algo y se tienen buenas razones para observar. Al observar se puede prestar especial atención tanto a detalles, como a lo sustancial y a los procedimientos. Observar lleva a descubrir cosas nuevas y es parte de un proceso de reaccionar significativamente ante el mundo. Al poner en práctica esta operación del pensamiento, se aprende a ver y reparar en lo que antes no se percibía, desarrollando así un criterio discriminativo que conduce a la maduración.

3.5.3. Aplicar hechos y principios a nuevas situaciones

Esta es una de las formas más comunes de enfatizar la importancia del pensamiento. En general, el individuo aprende ciertos principios, reglas, generalizaciones y leyes, y se familiariza con hechos fundamentales en un cierto contexto, para luego aplicarlos en otro distinto. Esto implica observar relaciones, advertir lo que debe “ir junto” a esta nueva situación, discriminar lo que es adecuado de lo que no es, y demás.

3.5.4. Tomar decisiones

Es semejante a la operación descrita previamente, pero hay una excepción importante. En la parte anterior se recalcó la importancia de las leyes, principios, generalizaciones y reglas. En este caso, las mismas no se omiten, pero en cambio se da mayor significado a la función de los *valores*. ¿Qué hacer y por qué? En este caso se presupone que el *por qué* revelará los valores que más aprecia el individuo.

Como puede observarse, la descripción de cada operación del pensamiento, al decir de Raths, es importante ya que describe detalladamente a qué refiere la operación y complementa la guía dada por la taxonomía. Si bien aquí no se han presentado todas las operaciones del pensamiento descritas en el libro de Raths, se han considerado aquellas que resultan operativas al diseño del juego.

3.6. Conclusiones del capítulo

En este capítulo se han presentado guías tales como la taxonomía de Bloom y su posterior revisión por Krathwohl como medios para la clasificación de los objetivos educativos. Si bien existen otras taxonomías, varias de ellas también constituidas como evoluciones de la taxonomía de Bloom, como es el caso de la taxonomía de Marzano (Marzano & Kendall, 2007), se ha decidido basarse en la de Krathwohl como guía en este caso para la definición de los objetivos del juego. Esta taxonomía ha sido elegida dado que los tesistas la han encontrado explicativa y operativa para el diseño del juego desde el punto de vista de los objetivos educacionales. Su utilización se pone en práctica en el capítulo 4 donde se aborda el diseño del juego que se presenta en este trabajo. Por otro lado, también se han mencionado las operaciones del pensamiento relacionadas al aprendizaje, las cuales también se consideran en el diseño del juego, ya que explican de manera más acabada los verbos propuestos como componente cognocitivo de la taxonomía de Krathwohl.

CAPÍTULO 4

Diseño de AstroCódigo

4.1. Introducción

En el presente capítulo se describen diferentes aspectos del diseño del juego desarrollado, cuyo nombre es AstroCódigo dada su temática espacial, la cual es descrita más adelante en este capítulo.

En las siguientes secciones se presenta la idea general de AstroCódigo, se detallan los objetivos educativos y se describen algunas decisiones generales tomadas en relación al desarrollo del juego. Luego se realiza una descripción completa de AstroCódigo detallando la historia propia del juego, su dinámica y los diferentes niveles por los que transita el jugador. Finalmente, se presenta una herramienta desarrollada para la generación de escenarios de programación en AstroCódigo que permite flexibilizar el juego y adaptarlo a diferentes situaciones educativas.

4.2. Descripción general de AstroCódigo

AstroCódigo pretende contribuir en el aprendizaje de conceptos básicos de programación. Para ello, plantea una temática de ciencia ficción, donde el jugador encarna a un astronauta programador argentino en una misión al espacio en la que explora diferentes planetas. La jugabilidad alterna entre 2 tipos: *a)* movimiento libre en primera persona en los distintos planetas en los que se desenvuelve la historia, y *b)* desafíos de programación (escenarios de programación), donde el jugador debe construir un algoritmo que permita a un determinado robot resolver el desafío. La construcción de estos algoritmos se realiza arrastrando y soltando bloques que representan instrucciones, condiciones y estructuras de control. El juego está enriquecido con una historia que busca captar la motivación de los jugadores, la cual es presentada en forma de cinemáticas entre los distintos planetas.

4.3. Objetivos educativos de AstroCódigo

Para la definición de los objetivos educativos se debe considerar primero cuáles son los conceptos básicos que se desea que el jugador aprenda con AstroCódigo. Entre ellos se destacan:

1. Secuencias de instrucciones.
2. Estructuras de control de decisión, repetición e iteración pre-condicional.
3. Variables.
4. Módulos.

El diseño del juego se realizó bajo la consideración de que permita abordar estos conceptos de diferentes maneras, ya sea a través de desafíos donde los jugadores deben programar algoritmos o a través de misiones secundarias en las que deben responder preguntas. En la dinámica de programación de algoritmos se abordan los dos primeros

conceptos (secuencias de instrucciones y el resto de las estructuras de control). En las misiones secundarias, las cuales se describen más en detalle en la sección 4.7 de este capítulo, se abordan los siguientes conceptos teóricos:

1. Lenguajes de programación.
2. Hardware y software.
3. Variables.
4. Módulos.

A continuación se presentan los objetivos educativos que se proponen alcanzar con el juego desarrollado. Para esto primero se enuncian los objetivos en forma de oración con el fin de identificar rápidamente sus respectivos verbos y sustantivos, y luego se los ubica en las intersecciones correspondientes de la tabla bidimensional de la taxonomía revisada de Bloom (ver Tabla 4.1) presentada previamente (Capítulo 2, sección 2.4.1).

a. Comprender el concepto de instrucciones simples y secuencias.

b. Construir algoritmos que resuelvan los distintos problemas presentados utilizando los conceptos previamente aprendidos sobre instrucciones simples y secuencias.

c. Identificar y seleccionar los recursos necesarios para la resolución de problemas.

d. Reconocer la necesidad de utilizar estructuras de control de decisión, repetición e iteración pre-condicional en la resolución de problemas.

e. Comprender el concepto de la estructura de control de decisión simple *si* (*if*), la de *repetir* (*for*) y la de iteración pre-condicional *mientras* (*while*).

f. Construir algoritmos que resuelvan los distintos problemas presentados, utilizando los conceptos previamente aprendidos de estructuras de control.

g. Reconocer y evaluar cuál es la solución óptima a un determinado problema.

h. Recordar conceptos teóricos básicos relacionados a los lenguajes de programación.

i. Recordar conceptos teóricos básicos relacionados al hardware y software y las diferencias entre estos.

j. Recordar conceptos teóricos básicos relacionados a variables.

k. Recordar conceptos teóricos básicos relacionados a módulos.

	1.Recordar	2.Entender	3.Aplicar	4.Analizar	5.Evaluar	6.Crear
A.Conocimiento Fáctico	h, i, j, k				h, i, j, k	
B.Conocimiento Conceptual		a, d, e	b, f, g	d, c		
A.Conocimiento Procedural				b, f	g	b, f
A.Conocimiento Metacognitivo						

Tabla 4.1: Objetivos educativos de AstroCódigo ubicados en la tabla bidimensional de la taxonomía revisada de Bloom.

4.4. Decisión sobre la narrativa o historia de ficción en AstroCódigo

A menudo, los estudiantes de informática pueden encontrar los temas relacionados a esta ciencia como difíciles o poco interesantes. En un estudio realizado por Waraich (2004), en el que se describe el diseño y construcción de una herramienta lúdica con el objetivo de motivar a alumnos en el estudio de estos temas, se hace hincapié en la necesidad de que los estudiantes encuentren la herramienta entretenida de usar, y para esto describe el uso de una historia o narrativa. El estudio establece que la inclusión de la narrativa en la enseñanza de conceptos abstractos y procedurales, como los vinculados a la programación, puede ser muy beneficioso para mejorar el entendimiento y motivación de los estudiantes.

Como ya se mencionó en esta tesina y en otros estudios, los juegos digitales posibilitan mantener la atención del jugador por largos periodos de tiempo, ya que permiten tomar riesgos en un mundo con pocas consecuencias negativas y proveen un contexto donde

probar habilidades de una manera entretenida y desafiante (Provenzo, 1991; Herz, 1997; Johnson, 1997; Prensky, 2001).

Malone (1981) ha identificado un número de razones por las cuales un juego digital es exitoso, entre ellas se encuentra el uso de una narrativa sencilla. Prensky (2001), también llega a las mismas conclusiones que Malone y sugiere que la narrativa puede ser una herramienta de motivación que puede producir fuertes emociones en el jugador.

La narrativa también puede ser vista como un mecanismo para darle sentido al material que se da, o como una parte fundamental del proceso de darle sentido. A su vez, se ha demostrado que la narrativa puede ser utilizada como un mecanismo para guiar al estudiante a través de un entorno de aprendizaje (Quinn, 1996). Como el objetivo de este trabajo es proveer un entorno de aprendizaje motivante para el alumno, resulta claro que contar con una historia o narrativa es una característica importante.

Luego de investigar diferentes formas utilizadas por los juegos para mostrar la historia que motiva al jugador, se decidió transmitirla mediante cinemáticas. La cinemática (en videojuegos) es un vídeo donde el jugador no tiene control o tiene un control limitado. Suele utilizarse para avanzar en una trama o proporcionar información. Así es que se diseñaron cinemáticas (ver Fig. 4.1) que se muestran en momentos claves del juego. Cada cinemática es acompañada por textos y buscan presentar de forma amena y clara la historia, favoreciendo la inmersión del jugador en el ambiente que se desea construir.

Figura 4.1: Ejemplo de cinemática.

El argumento presentado en las cinemáticas se detalla a continuación.

4.4.1. Argumento

Corre el año 2458 y el Comando de Exploración Galáctica (CEG) en la Tierra, utilizando el satélite *Milanesat Conpuret*, capta ondas misteriosas provenientes de un planeta desconocido en una galaxia cercana. Se le ha encomendado a un astronauta, cuyo nombre clave asignado para la misión es Ankatu (nombre mapuche que significa “el señor que ha tocado el cielo”), la misión de llegar a este planeta inexplorado para descifrar el misterio que allí yace.

En el camino a su destino, un malfuncionamiento en la nave obliga a Ankatu a realizar un aterrizaje forzoso en un planeta desconocido. Antes de impactar en el planeta, es eyectado de su nave en una cápsula de salvamento. Ahora debe recolectar las piezas de la nave, para ello debe atravesar distintos escenarios de programación que encontrará dispersos en el planeta, recolectando piezas en cada uno. Finalmente debe llegar a las ruinas de su nave y

analizar qué es lo que puede hacer para recuperarla.

Una vez reconstruida su nave, el astronauta continúa su viaje hasta el planeta a donde originalmente se dirigía, pero en el camino sufre un ataque directo de piratas espaciales, los cuales aparecen para advertirle que no continúe con su misión o deberá atenerse a las consecuencias. Nuevamente Ankatu se ve forzado a aterrizar en otro planeta desconocido para recolectar las partes de su nave dañada y continuar con su misión, ahora con un temible enemigo en su camino.

Una vez revisados los restos de la nave, no cabe ninguna duda para el astronauta de que estos piratas quieren evitar a toda costa que continúe la exploración de la misteriosa señal. A su vez, el hecho de que exista tanto esfuerzo para evitar que cumpla su misión le indica al astronauta que el secreto de la señal puede ser aún más valioso de lo que se imagina.

Ankatu repara su nave y se dirige guiado por la extraña señal a otro planeta no muy lejano a su última ubicación. A partir de la exploración de este nuevo mundo, descubre un templo erigido en la cima de una montaña formada por un volcán. No cabe duda para Ankatu, que allí es donde se encuentra la fuente de la misteriosa señal, pero al parecer los piratas se le han adelantado y han entrado en el templo antes que él.

Dentro del templo, se encuentra con antiguos y perdidos conocimientos de la programación escritos en los muros. Y en el nivel más alto, descubre que los piratas intentan alcanzar un antiguo libro que contiene los secretos más profundos de la programación. Este libro es de un valor incalculable, por ello es el deseo de los piratas robarlo y venderlo en el mercado negro. Habiendo intentado sin éxito alcanzar el libro los piratas entienden que por separado les es imposible, pero junto con Ankatu podrán lograrlo y deciden compartir su sabiduría. Así el astronauta y los piratas terminan cooperando entre sí para luego leer juntos el libro. Terminada su misión, Ankatu vuelve a la tierra y es recibido como un héroe dando final al juego.

4.5. Escenarios de programación

Los escenarios de programación son las instancias en las que el jugador aprende o pone a prueba sus conocimientos de programación. En cada planeta se pueden encontrar múltiples escenarios. Cada escenario presenta un mapa con piezas. El jugador debe escoger un robot para recorrer ese mapa y armar un algoritmo que permita al robot escogido obtener todas las piezas del mapa. A lo largo del juego se encuentran distribuidos diez escenarios de programación, donde cada uno presenta una dificultad diferente (ver sección 4.6). Al iniciar cada escenario, se presenta una serie de ayudas para introducir los conceptos y dificultades propias del escenario (ver sección 4.5.2). El jugador debe *aplicar* los conocimientos adquiridos en estas ayudas para resolver el escenario. La primera *decisión* que debe tomar el jugador al iniciar un escenario, es seleccionar el robot indicado para resolverlo (relacionado con el objetivo educativo c, especificado en la sección 4.3). Para esto se dispone de un panel que permite *observar y comparar* las características de cada robot. A continuación se explica la interfaz de estos escenarios y la forma en la que el jugador arma el programa para resolverlos.

En esta subsección se explica entonces: la interfaz, los tutoriales, los robots, y la forma de dar el *feedback* a lo largo de los diferentes escenarios de programación.

4.5.1. Interfaz

La interfaz de los escenarios de programación se divide en cuatro grandes secciones (ver Fig. 4.2): mapa del escenario, consola de programa, menú lateral, y panel de programación. El mapa del escenario, ubicado en el centro de la pantalla, representa el terreno por el cual se debe mover al robot para obtener todas las piezas objetivo (ver Fig. 4.8f). El jugador debe observar los casilleros de dicho mapa y sus disposiciones para luego determinar qué robot utilizar y cómo construir el programa.

Figura 4.2: Interfaz de los escenarios de programación.

En la parte inferior derecha de la pantalla se ubica la consola y sirve para mostrar mensajes relacionados a la construcción y ejecución del programa. Los mensajes mostrados en la consola pueden ser de error (rojos), de advertencia (amarillos) o informativos (verdes). En el borde derecho de la pantalla se encuentra el menú lateral, el cual cuenta con botones para cambiar el modo de cámara para ver el escenario completo o para seguir al robot más de cerca, para ingresar al índice de tutoriales, para ver la insignia conseguida en el escenario, y para salir del escenario. Por último, en la sección izquierda de la pantalla se encuentra el panel de programación, que a su vez se divide en tres subpaneles: subpanel de robot, subpanel de instrucciones y subpanel de programa. El subpanel de robot permite ver y cambiar el robot seleccionado. Por otro lado, el subpanel de instrucciones muestra todas las instrucciones, estructuras de control y condiciones disponibles con las que cuenta el jugador para construir el programa. Todo lo que se encuentre en el subpanel de instrucciones puede ser arrastrado y soltado al subpanel de programa, donde se va construyendo el algoritmo que solucionará el problema del escenario. En la parte inferior del subpanel de programa se encuentran los botones que permiten controlar el inicio y parada de la ejecución, la velocidad del robot y el borrado del programa.

4.5.2. Tutoriales

Al iniciar un escenario de programación, lo primero que se muestra son tutoriales que ayudan a comprender los nuevos conceptos y elementos incorporados en cada escenario. Existen tutoriales tanto para explicar la interfaz general de los escenarios como para las características y funcionamientos de cada robot, casillero y estructura de control. Si bien al comienzo de cada escenario solo se muestran algunos tutoriales, el jugador puede acceder a todos los tutoriales desde el menú derecho. Cada tutorial se compone de distintas secciones las cuales a su vez incluyen un video explicativo asistido por audio y texto.

4.5.3. Robots

A lo largo del juego, los escenarios de programación no solo van creciendo en dificultad, sino que también presentan nuevos obstáculos que el jugador debe enfrentar. Los obstáculos requieren de diferentes habilidades, por ello el juego cuenta con cuatro robots distintos, cada uno con capacidades diferentes. En cada escenario el jugador debe *observar* los obstáculos presentes en el mismo y *comparar* entre los robots disponibles para seleccionar el adecuado y poder resolver el escenario.

Cada robot dispone de un set de instrucciones disponibles. Habrá algunas que serán comunes entre todos ellos, y por ende están siempre disponibles, como son las de movimiento simple (avanzar, derecha, izquierda, terminar y agarrar). Pero existen otras específicas de cada robot que están disponibles sólo cuando se utilice el robot correspondiente. Éstas le sirven para ejecutar tareas específicas necesarias para completar un nivel en particular.

A continuación de describen los robots con sus habilidades:

Explorador

El Explorador es el robot inicial con el que el jugador se introduce al juego y está disponible desde el inicio del mismo (ver Fig. 4.3). Solo posee las instrucciones básicas de movimiento, siendo estas:

- ***Avanzar***, el robot avanza un casillero en dirección hacia donde esté orientado;
- ***Derecha***, el robot gira sobre su propio eje en sentido horario;
- ***Izquierda***, el robot gira sobre su propio eje en sentido anti-horario;
- ***Agarrar***, el robot intenta tomar una pieza en el casillero ubicado inmediatamente delante al robot (cabe destacar que si se intenta agarrar una pieza donde no la hay, el programa terminará con un error fatal);
- ***Terminar***, indica el final del programa a ejecutar por el robot y ubica al robot devuelta en el casillero inicial.

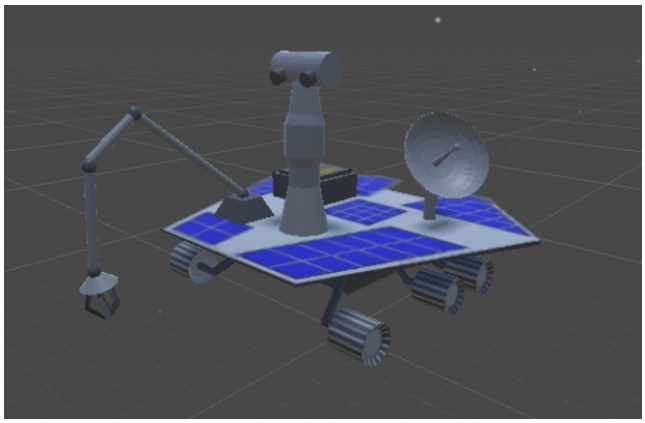

Figura 4.3: Robot Explorador.

Laserbot

El robot Laserbot (ver Fig. 4.4) tiene, además de las instrucciones básicas de movimiento, instrucciones que le permiten destruir bloques de pared (Fig. 4.8b) que se interpongan en

su paso. Para obtener este robot se debe resolver el último escenario del primer planeta. En particular, este robot incorpora la instrucción:

- ***Destruir***, que permite, como su nombre lo indica, destruir un casillero pared que esté ubicado inmediatamente frente a él.

Figura 4.4: Robot Laserbot.

Drone

El robot Drone (ver Fig. 4.5) tiene la capacidad de volar, lo cual le permite abrirse camino entre obstáculos siendo esto imposible para otros robots. Este robot se obtiene al resolver el segundo escenario del segundo planeta. Además de las instrucciones básicas de movimiento, el robot cuenta con la habilidad de abrir los casilleros cofres simplemente posicionándose sobre ellos. Este robot también provee:

- ***HayPieza***, condición que puede ser utilizada dentro de las estructuras de control disponibles en ciertos escenarios para determinar si el casillero inmediatamente debajo del Drone posee o no una pieza. Esta condición debe ser utilizada también cuando se desee recoger una pieza dentro de un cofre ya que los cofres pueden estar vacíos.

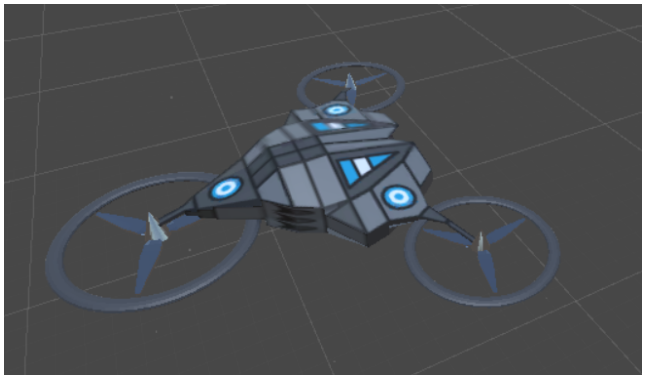

Figura 4.5: Robot Drone.

Perrobot

El Perrobot (ver Fig. 4.6) tiene la capacidad de desenterrar piezas. Este robot se consigue al resolver el cuarto escenario del segundo planeta y es el último robot del juego. El robot Perrobot, incorpora las instrucciones:

- ***Excavar***, que le permite remover tierra de un casillero tierra. Los casilleros tierra contienen una cantidad aleatoria de tierra, por lo que el robot posiblemente tendrá que utilizar la instrucción excavar más de una vez. Excavar sólo realiza la acción una vez.
- ***HayTierra***, condición que determina si el casillero debajo del robot posee tierra o no. Esta condición puede ser utilizada en conjunto con la estructura de control *mientras* para excavar toda la tierra que posea un casillero.
- **HayPieza**, condición que permite verificar si el casillero tierra, luego de ser excavado, posee una pieza o no.

Figura 4.6: Robot Perrobot.

4.5.4. Feedback

Al resolver exitosamente un escenario de programación, al jugador se le provee una retroalimentación sobre la calidad del algoritmo construido, la cual se presenta en forma de insignia. Si el jugador no consigue la mejor insignia, es decir, si el algoritmo no es óptimo, además de la insignia se proveen consejos para mejorar la calidad de la solución. La insignia otorgada se calcula en base a varios factores entre los cuales influye la cantidad de bloques de instrucciones utilizadas, la cantidad de instrucciones ejecutadas por el robot, la utilización de las estructuras de control, la verificación de todos los casilleros cofre o tierra, y la cantidad de advertencias conseguidas en la consola. Las insignias disponibles (ver Fig. 4.7) en todos los escenarios son, de menor mérito a mayor: *insignia de programador principiante*, *insignia de programador intermedio*, *insignia de programador avanzado* e *insignia de programador intergaláctico*. Adicionalmente, en ciertos escenarios donde el jugador identifique un patrón avanzado de movimientos haciendo uso de estructuras de control, se dispone también de la *insignia de programador legendario*, siendo ésta la de mayor mérito en estos casos.

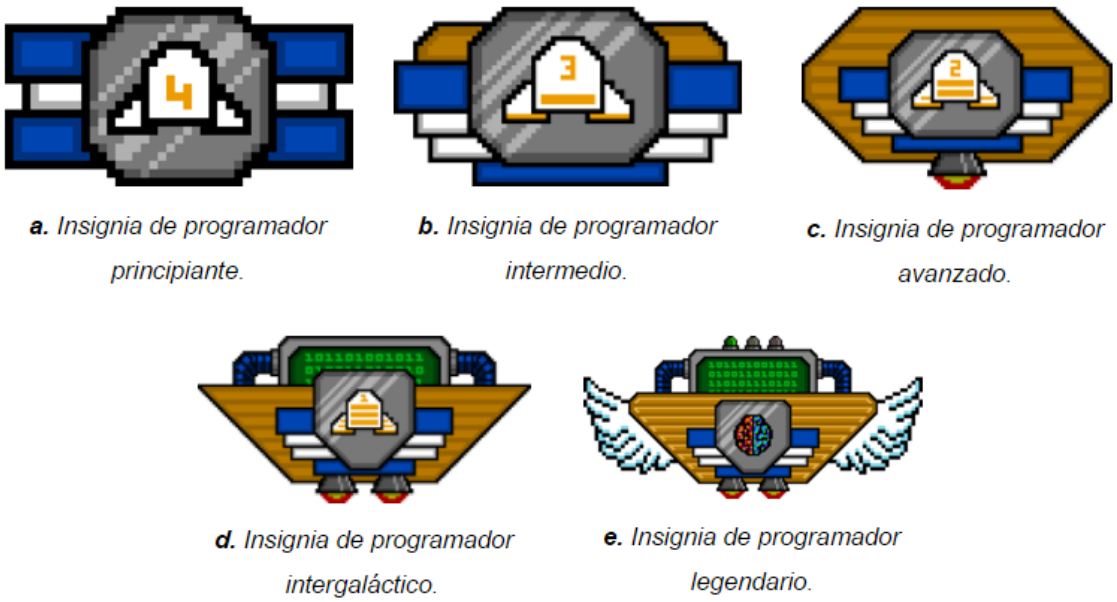

Figura 4.7: Insignias disponibles que representan la calidad de las soluciones construidas por el jugador para los escenarios de programación.

La decisión de incorporar esta retroalimentación en los escenarios de programación, fue tomada con la intensión de motivar al jugador a aprovechar al máximo los aprendizajes que subyacen en la resolución óptima de cada escenario (Seel, 2012). La búsqueda de la solución óptima, permite que el jugador se preocupe por encontrar patrones de movimiento y así aplicar estructuras de control en forma adecuada, familiarizándose en su utilización (esto se relaciona con el objetivo g del juego, especificado en la sección 4.3). Además, en base a las insignias conseguidas en todos los escenarios, se calcula el puntaje final del jugador, que luego lo posiciona en el ranking que se muestra en el sitio web. Esto último intenta generar una actitud competitiva en los jugadores y fomentar una mayor motivación en conseguir las mejores insignias construyendo soluciones óptimas en cada escenario.

4.6. Planetas y escenarios

Como ya se presentaron en la sección 4.5 los escenarios son las instancias en las cuales los jugadores crean algoritmos. Los escenarios se encuentran a lo largo de la exploración de los distintos planetas a donde el astronauta llega. Cada planeta representa un nivel del juego y los escenarios ubicados en cada uno van aumentando la complejidad requerida en los algoritmos a desarrollar. Se pretende así que el jugador ponga en práctica algunas de las operaciones cognitivas de orden superior definidas por Raths (1967) tales como la observación, comparación, toma de decisiones y la capacidad de aplicar hechos y principios a nuevas situaciones.

A continuación se explica cómo se conforman los escenarios. Luego, se detallan los planetas y los escenarios de todo el juego, teniendo en cuenta los conceptos de programación involucrados en cada uno, los objetivos educacionales y las particularidades de cada escenario.

4.6.1. Diccionario de bloques de escenario

La Figura 4.8 muestra los diferentes bloques que conforman los distintos mapas de los escenarios a lo largo del juego. Estos son: casillero vacío (Fig. 4.8a), casillero pared (Fig. 4.8b), casillero tierra (Fig. 4.8c), casillero cofre (Fig. 4.8d), casillero agua (Fig. 4.8e) y casillero pieza (Fig. 4.8f). La selección y distribución de estos bloques determinan la complejidad del algoritmo necesario para recorrer el mapa y resolver el escenario de programación.

Figura 4.8: Casilleros que pueden encontrarse en los mapas de los escenarios de programación.

A continuación se detallan los planetas con sus escenarios. En cada escenario pueden notarse los diferentes diseños de mapas y su complejidad.

4.6.2. Planeta 1

En el primer planeta del juego, se introduce al jugador a las mecánicas básicas del juego. Este planeta tiene una geografía montañosa y posee un camino que guía al jugador hacia los escenarios esparcidos por el mismo (ver Fig. 4.9). Estos escenarios presentan una dificultad baja para luego ir escalando en complejidad y dificultad en los escenarios del segundo planeta. Adicionalmente, este planeta cuenta con dos misiones secundarias las cuales abordan los temas relacionados a lenguajes de programación y diferencias entre hardware y software.

Para resolver inicialmente los escenarios, el jugador solo cuenta con el robot Explorador. Los demás robots se adquieren a medida que se resuelven escenarios.

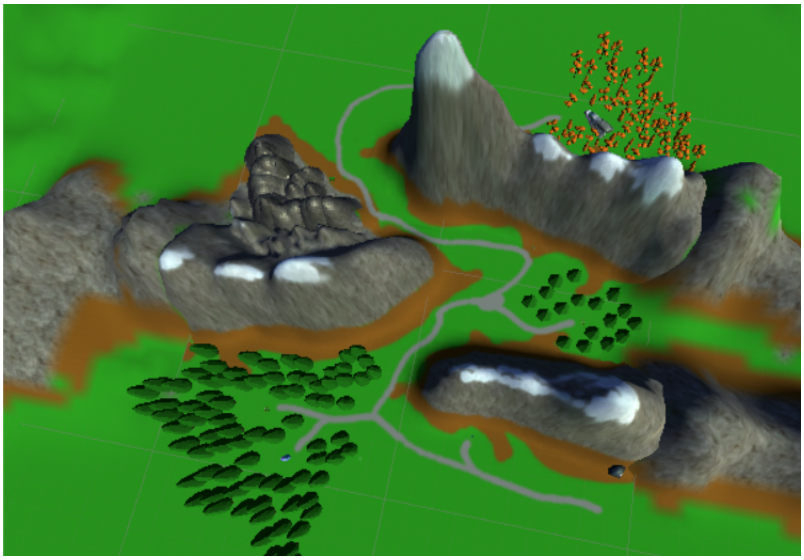

Figura 4.9: Imagen aérea del planeta 1.

Escenario 1

Este escenario es muy simple y fácil de resolver, la idea del mismo es presentar una introducción a la mecánica del juego (ver Fig. 4.10). No contiene ningún tipo de obstáculo y el jugador solo debe conseguir una pieza. En este escenario se pretenden cumplir los objetivos educativos *a*, *b* y *c* especificados en la sección 4.3.

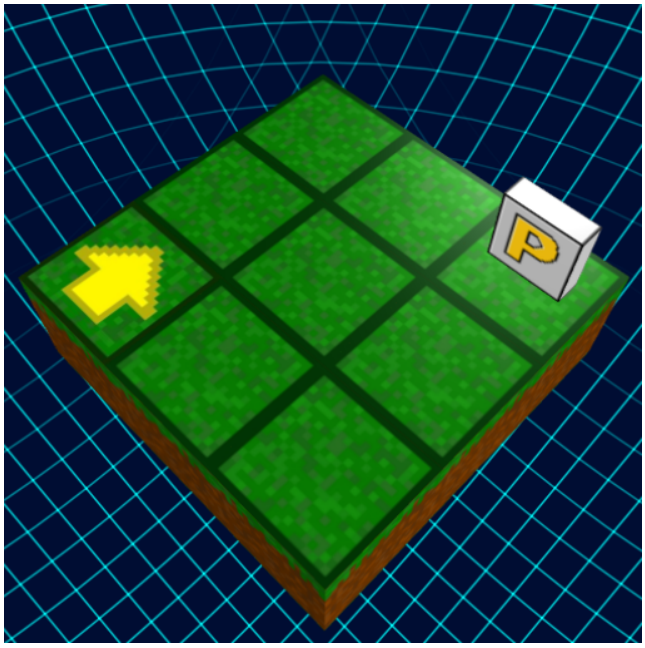

Figura 4.10: Escenario 1.

Escenario 2

Este escenario introduce los obstáculos correspondientes a los casilleros *pared* y *agua*, por lo que presenta una dificultad ligeramente superior al escenario anterior, ya que para obtener las dos piezas disponibles el jugador debe programar al robot para sortear estos obstáculos (ver Fig. 4.11).

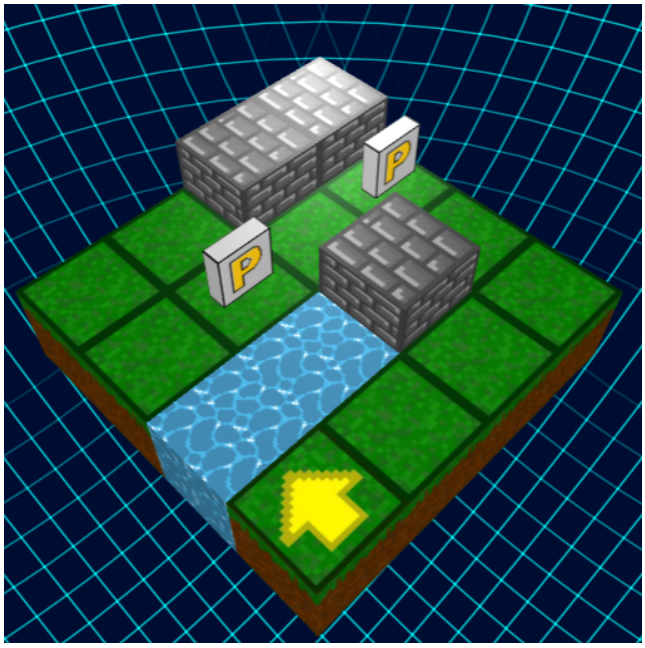

Figura 4.11: Escenario 2.

Escenario 3

Este escenario presenta nuevamente una dificultad ligeramente superior al anterior y cuenta con los mismos obstáculos (pared y agua), pero en mayor cantidad (ver Fig. 4.12). Al igual que en los niveles anteriores el jugador debe sortear los obstáculos para recolectar las piezas objetivo. Este escenario plantea la situación de instrucciones iguales contiguas introduciendo así la estructura *repetir* (*for*) para simplificar los algoritmos desarrollados. Aquí se pretenden cumplir los objetivos educativos f y g especificados en la sección 4.3.

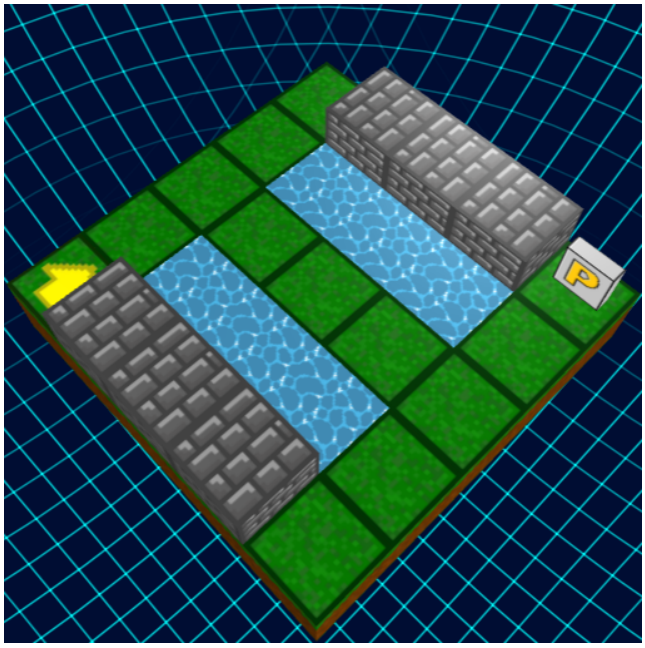

Figura 4.12: Escenario 3.

Escenario 4

Este escenario, al ser el último de este planeta, presenta la mayor dificultad que se puede lograr con las instrucciones básicas de movimiento. Cuenta con los mismos obstáculos (pared y agua) pero en mayor cantidad (ver Fig. 4.13). Al igual que en los niveles anteriores, el jugador debe sortear los obstáculos para recolectar las piezas objetivo. Al resolver este escenario, el jugador obtiene el robot *Laserbot*, y puede utilizarlo en cualquier escenario.

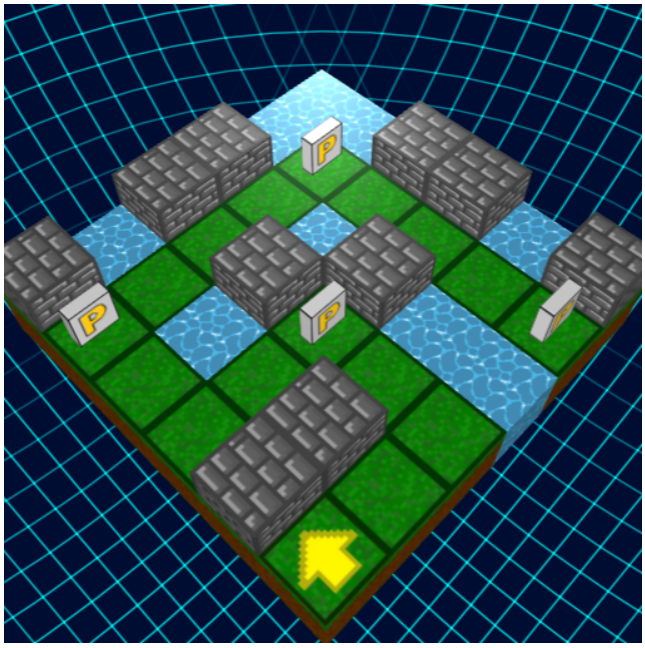

Figura 4.13: Escenario 4.

4.6.3. Planeta 2

El segundo planeta del juego tiene una extensión más grande y entre sus montañas y ríos se esconden los seis escenarios de programación restantes y dos misiones secundarias que introducen teóricamente los conceptos de variables y módulos (ver Fig. 4.14). Los escenarios de este planeta escalan significativamente en dificultad a medida que se van presentando nuevos robots, casilleros especiales y estructuras de control. Al igual que en el planeta anterior, hay un camino que el jugador puede utilizar para guiarse y llegar a los distintos puntos clave del planeta.

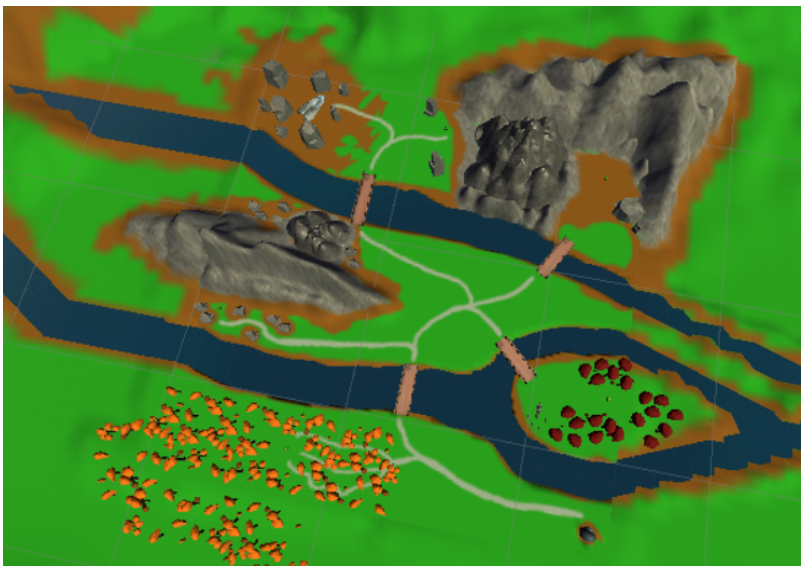

Figura 4.14: Imagen aérea del planeta 2.

Escenario 5

El principal propósito de este escenario es el de introducir al jugador al uso del robot *Laserbot* recientemente adquirido, por lo que no presenta grandes dificultades y su resolución es simple (ver Fig. 4.15). Aquí el jugador debe utilizar la instrucción especial *destruir* provista por el robot *Laserbot*. Este escenario sólo puede ser resuelto con los robots *Laserbot* y *Drone*.

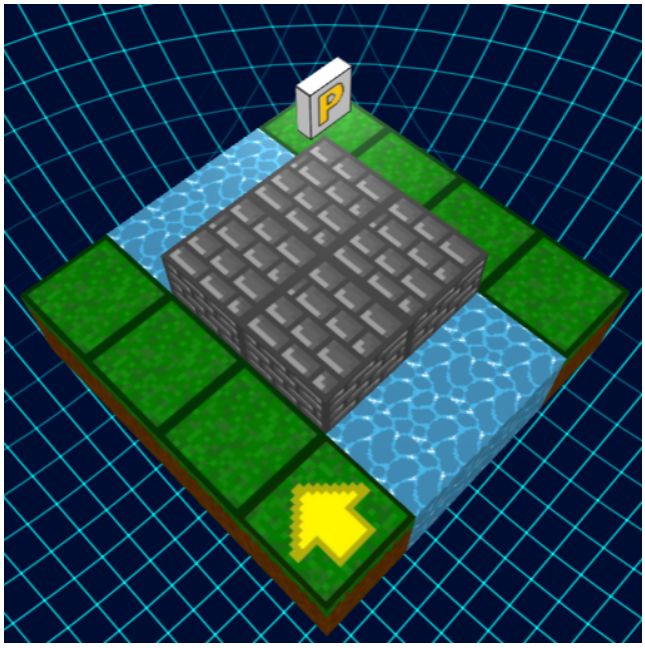

Figura 4.15: Escenario 5.

Escenario 6

Este escenario es similar al anterior en cuanto a que el jugador se ve obstaculizado por casilleros pared para obtener las piezas (ver Fig. 4.16). A diferencia del anterior, este escenario presenta una dificultad ligeramente superior ya que tiene tamaño de mapa mayor y una mayor cantidad de obstáculos que deben ser destruidos para alcanzar las piezas objetivo. Al resolver este escenario, el jugador obtiene el robot *Drone*, y puede utilizarlo en cualquier escenario.

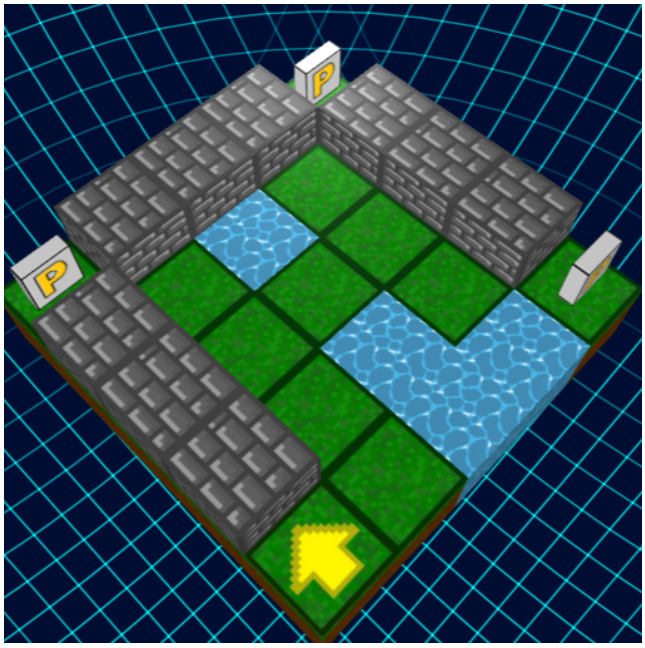

Figura 4.16: Escenario 6.

Escenario 7

Este escenario, aunque más pequeño, presenta una complejidad superior al anterior, dado a que incorpora los casilleros *cofre*, los cuales pueden o no contener una pieza objetivo (ver Fig. 4.17). Para resolver este escenario el jugador deberá utilizar el robot *Drone* ya que es el único robot con la capacidad de sobrevolar los cofres, además el jugador debe hacer uso de la estructura de control *si* (*if*), la cual es introducida en este escenario, para verificar si un cofre contiene una pieza.

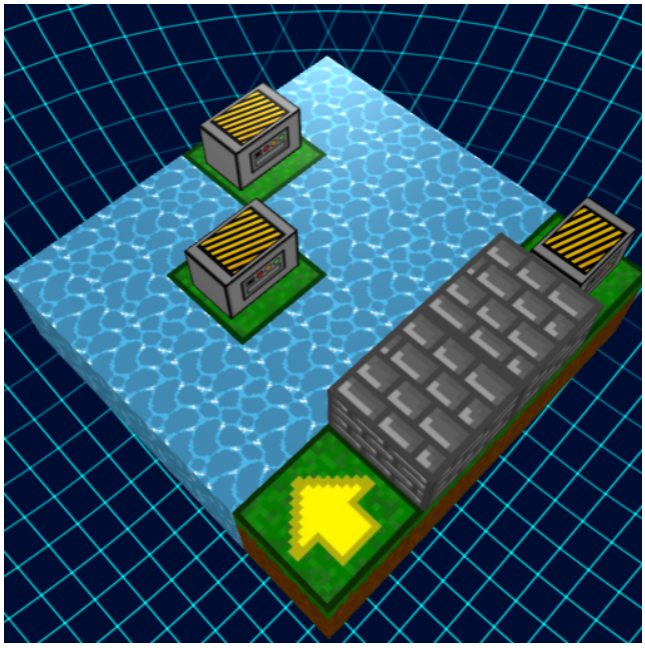

Figura 4.17: Escenario 7.

Escenario 8

Este escenario presenta los mismos obstáculos a sortear que el anterior, pero es de un tamaño mayor y con un número mayor de cofres, de esta forma el jugador consolidará los conocimientos obtenidos en el escenario anterior (ver Fig. 4.18). Al resolver este escenario el jugador obtiene el robot *Perrobot*.

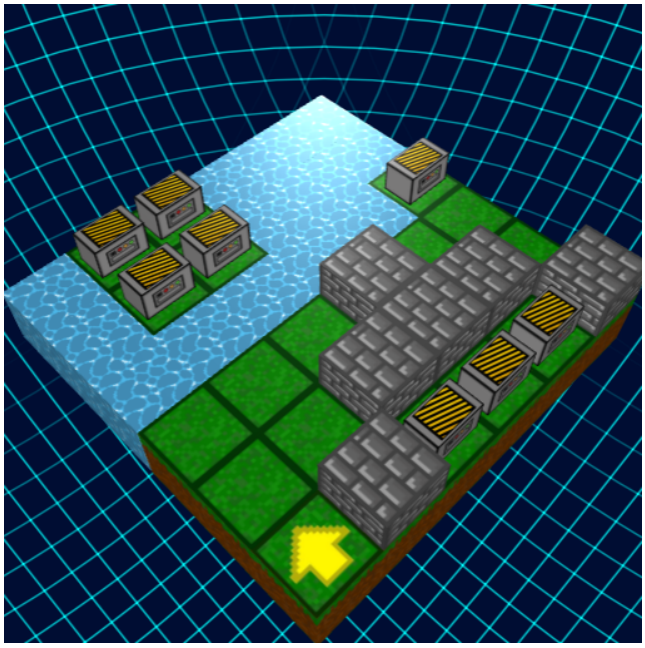

Figura 4.18: Escenario 8.

Escenario 9

Este escenario nuevamente presenta una dificultad diferente al anterior, dado que es necesario utilizar el robot *Perrobot* y su instrucción especial *excavar* en los casilleros *tierra*, dentro de los cuales puede haber piezas objetivo (ver Fig. 4.19). Una de las peculiaridades del bloque *tierra*, es que puede requerir más de una instrucción de *excavar* para revelar su contenido, por lo que se introduce la estructura de control *mientras* (*while*). Este escenario, al igual que todos los que contengan casilleros *tierra*, sólo pueden ser resueltos por el robot *Perrobot*.

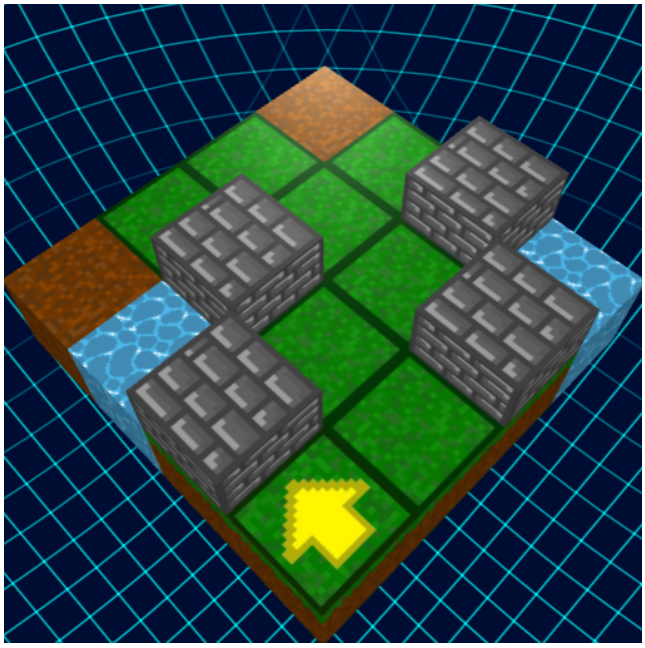

Figura 4.19: Escenario 9.

Escenario 10

Este escenario presenta las mismas características que el anterior, pero es de un tamaño mayor y con un número mayor de casilleros *tierra*, con el objetivo de consolidar los conocimiento adquiridos en el escenario anterior (ver Fig. 4.20).

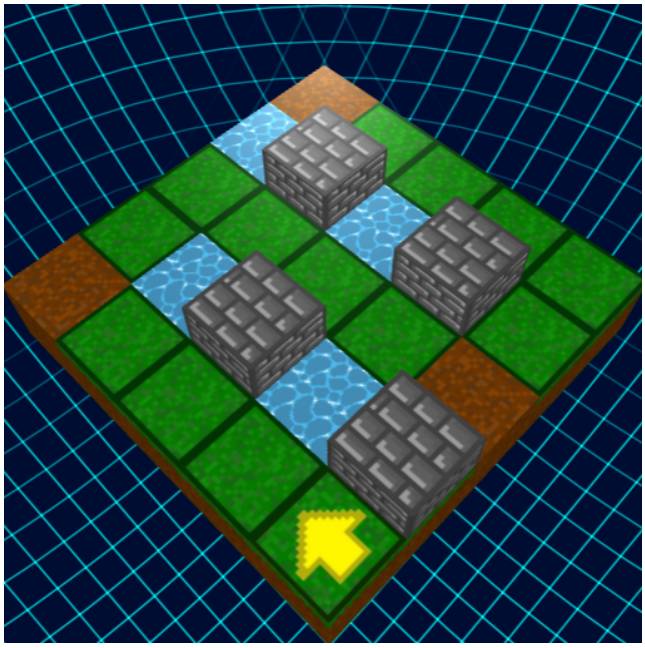

Figura 4.20: Escenario 10.

4.6.4. Planeta 3

Con el objetivo de introducir un cierre a la historia que se desarrolla en el juego se creó el Planeta 3 (ver Fig. 4.21). Este no cuenta con escenarios de programación, sino con locaciones en el mapa que se recorre en primera persona, donde el jugador debe resolver una misión muy simple para descubrir el final de la historia del juego.

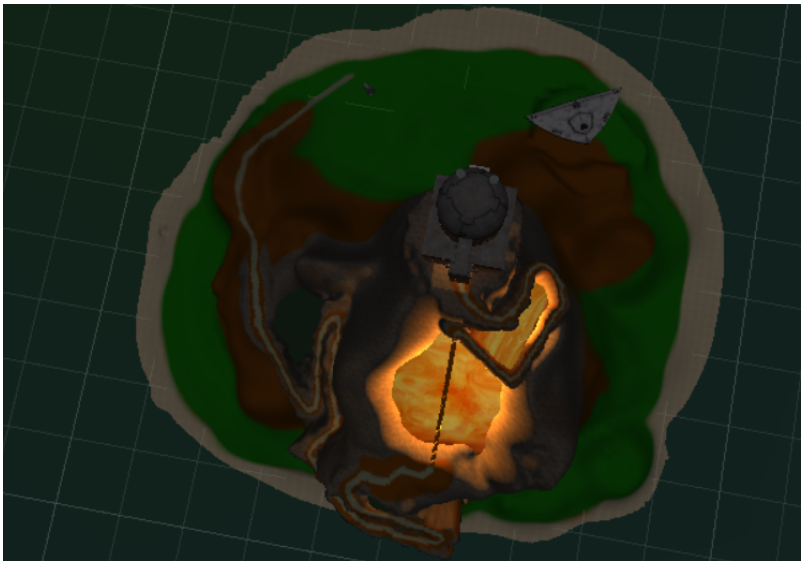

Figura 4.21: Imagen aérea del planeta 3.

4.7. Misiones secundarias

Con el objetivo de incluir conceptos teóricos relacionados a la Informática y la programación, se ideó un sistema de misiones secundarias dentro del juego. Estas misiones se encuentran distribuidas en el área de los planetas 1 y 2, por lo tanto pueden ser encontradas sólo en la modalidad de primera persona. Cada misión secundaria está guiada por un extraterrestre. El extraterrestre inicialmente muestra una pequeña introducción en forma de texto y solicita al astronauta que busque un escrito que se encuentra en las cercanías, por lo general plasmado en una estructura o monolito. Este escrito presenta conceptos sobre Informática, como un extracto correspondiente a un hecho histórico de la programación, alguna particularidad de los lenguajes, definiciones sobre conceptos relacionados a la programación, etc. Luego de leer el escrito el jugador debe regresar al punto donde se encuentra el extraterrestre y responder una serie de preguntas seleccionando de entre un conjunto de opciones (tipo *multiple choice*). Al finalizar, el extraterrestre informa si la misión ha tenido éxito o no. En caso de haber respondido todas las preguntas correctamente, el extraterrestre felicita, agradece e informa que se ha completado la misión. En caso de haber respondido

alguna pregunta de manera incorrecta, el extraterrestre recomienda regresar a leer el texto de la estructura o monolito y realizar el cuestionario nuevamente.

Ya que, como los escenarios de programación, las misiones secundarias poseen un valor educativo dentro del juego, es importante que completarlas también tenga una recompensa en puntaje para el jugador. Dado que sólo pueden ser superadas contestando todas las preguntas del extraterrestre de manera correcta, el puntaje que otorgan será constante.

Como una forma de ayudar a la jugabilidad, los extraterrestres cuentan con un símbolo en 3D encima de ellos, que indica de una manera simple y rápida el estado de la misión que el mismo otorga. Cuando la misión no ha sido superada se muestra un signo de exclamación, y en caso contrario el signo estará ausente.

4.8. Decisiones generales

En esta sección se ahondará sobre diferentes decisiones que surgieron en la etapa de diseño del juego, y cómo fueron afrontadas y solucionadas.

4.8.1. Sitio web

En el afán de llegar a la mayor cantidad de jóvenes posibles, se construyó un sitio web (http://www.astrocodigo.com/) en el cual es posible descargar el juego en sus diferentes versiones correspondientes a los sistemas operativos que soporta (Windows, Linux, MacOsX), encontrar información del juego e información sobre los desarrolladores y el ámbito institucional.

Además, para promover un espíritu de sana competencia entre los jugadores y motivarlos a mejorar sus puntajes (y así sus algoritmos en el juego), se decidió confeccionar un ranking de puntajes, el cual se muestra en una sección del sitio web. Para poder acceder al *ranking* es necesario que los jugadores se registren en el sitio web y accedan al juego mediante los

datos de esta cuenta. Luego, durante el juego, se realiza una conexión con la base de datos de AstroCódigo y se actualizan los datos correspondientes a la cuenta utilizada como son el puntaje, datos de progreso y última ubicación dentro del juego, de manera tal que no se pierdan los avances del jugador.

En el sitio web también se encuentra un apartado que contiene la herramienta para diseñar escenarios personalizados. Esta herramienta es totalmente web y permite construir y personalizar escenarios que luego pueden ser accedidos dentro del juego. Esta herramienta se explica con más detalle hacia el final de este capítulo.

4.8.2. Plataforma

En un comienzo se planificó que el juego corriera completamente sobre una plataforma web, y fuera parte del sitio antes mencionado, de tal manera que fuera accesible para cualquier persona en cualquier lugar que tuviera un navegador web y acceso a internet. Pero luego de avanzado el desarrollo se encontraron inconvenientes que dificultaron la implementación web de la herramienta, como por ejemplo el excesivo consumo de memoria RAM del motor gráfico WebGL. Por estas razones se decidió mudar el desarrollo a un ejecutable compilado específicamente para cada plataforma, Windows, Linux y OSX. Como se mencionó antes este ejecutable se puede descargar desde el sitio web del juego.

4.9. Generador de escenarios personalizados

A continuación se describe el funcionamiento del generador de escenarios que permite diseñar nuevos mapas y visualizar el camino que debe realizar el algoritmo para el mismo. El desarrollo del generador de escenarios personalizados nació de la necesidad de flexibilizar el juego para permitir que sea adaptado por los educadores a distintas situaciones. Así, se imaginó una herramienta sobre la cual se pudiera diseñar y construir nuevos escenarios de programación, y de esa manera definir nuevos objetivos y desafíos que no se encuentran en

el juego original.

El generador se desarrolló como una herramienta web disponible desde cualquier computadora que disponga de un navegador y acceso a internet. Esta herramienta trabaja en conjunto con una base datos online, que permite guardar los escenarios creados y que luego sean recogidos por el juego para ser mostrados en la sección de escenarios personalizados del menú inicial.

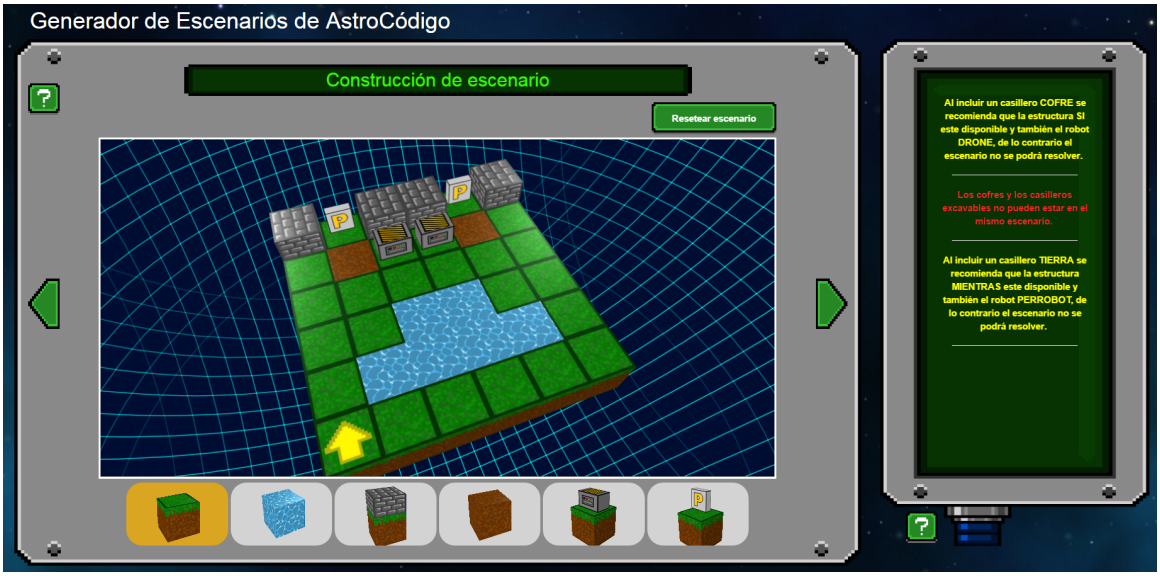

Figura 4.22: Generador de escenarios.

El apartado gráfico del generador se asimila al juego para lograr que ambos se vean como partes de una misma herramienta educativa. Por esto, se decidió que los errores se presenten en una consola similar a la que se muestra en el juego, y que los contenedores de la web tuvieran fondos similares a los que utiliza el juego (ver Figura 4.22).

El generador de escenarios tiene una importante funcionalidad que consiste en el cálculo de los caminos mínimos. Utilizando como base el algoritmo de Dijkstra (Aho, 1988) para el cálculo de caminos mínimos en un grafo, se pensó en encontrar los mejores caminos que pudiera recorrer el robot para recoger cada pieza y así construir una aproximación al mejor camino posible para recoger todas las piezas del nuevo escenario creado con el generador.

El cálculo del camino mínimo se utiliza en esta herramienta con dos objetivos. Por un lado, para mostrar gráficamente el mejor camino cuando se crea un nuevo escenario y, a su vez, armar y mostrar el mejor programa que podría construir el jugador para recolectar todas las piezas. Por otro lado, con el cálculo de la cantidad de instrucciones ejecutadas y escritas del mejor programa posible, se determinan las insignias (ver sección 4.5.4) correspondientes y los puntajes dentro del juego. Mientras más cerca del algoritmo óptimo es el algoritmo creado por el jugador, mejor será la insignia otorgada y por lo tanto el puntaje final.

4.10. Conclusiones del capítulo

A lo largo de este capítulo se han mostrado los distintos aspectos de diseño de AstroCódigo, incluyendo los objetivos educativos propuestos, las distintas mecánicas de juego, la descripción de los planetas y escenarios, los robots disponibles, y otros aspectos importantes circundantes al juego mismo, tales como su sitio web y la generación de niveles personalizados. En el siguiente capítulo se abordarán los aspectos relacionados a la implementación y las dificultades encontradas al desarrollar el juego.

CAPÍTULO 5

Aspectos relacionados a la implementación de AstroCódigo

5.1. Introducción

En este capítulo se describe el proceso realizado para alcanzar la implementación de AstroCódigo, incluyendo una descripción de las tecnologías utilizadas, los motivos de la elección de las mismas, los desafíos y problemas enfrentados a lo largo del proceso, así como las decisiones y cambios de rumbo que se sucedieron. El capítulo está estructurado de tal forma que se describe lo anteriormente mencionado por cada uno de los grandes componentes del juego, brindando detalles técnicos sobre la implementación de estos.

5.2. Decisiones vinculadas al desarrollo del juego

Para el desarrollo del juego, se optó por utilizar *Unity 3D* versión 5.3.5f1, la cual era la última versión estable al momento del comienzo del desarrollo. *Unity 3D* es un motor de videojuegos que provee un entorno de desarrollo amigable, permitiendo la creación de

juegos de forma rápida y simple. Una de las ventajas que presenta esta tecnología es su gran comunidad de desarrolladores, que además de crear tutoriales y guías que ayudan al aprendizaje, crean *assets* que luego pueden ser adquiridos en la *Unity Asset Store* para su reutilización en juegos propios. Otras de las grandes ventajas de este motor de juegos, es la gratuidad tanto de su uso como a la hora de compilar los proyectos a las múltiples plataformas que soporta. Estas características, sumadas al conocimiento previo en el uso de *Unity 3D* por parte de los tesistas, fueron determinantes a la hora de decidir la tecnología a utilizar para el desarrollo de AstroCódigo.

El lenguaje de programación utilizado fue C#, por lo que se hizo uso del paradigma de Programación Orientada a Objetos. En la Fig. 5.1 se muestra un diagrama de clases que grafica la forma en que se implementaron los escenarios de programación. *Unity* brinda también la posibilidad de utilizar JavaScript como lenguaje de programación, pero se optó por C# por su mayor popularidad en comparación a su contraparte en cuanto a *assets* disponibles en este lenguaje, guías y documentación.

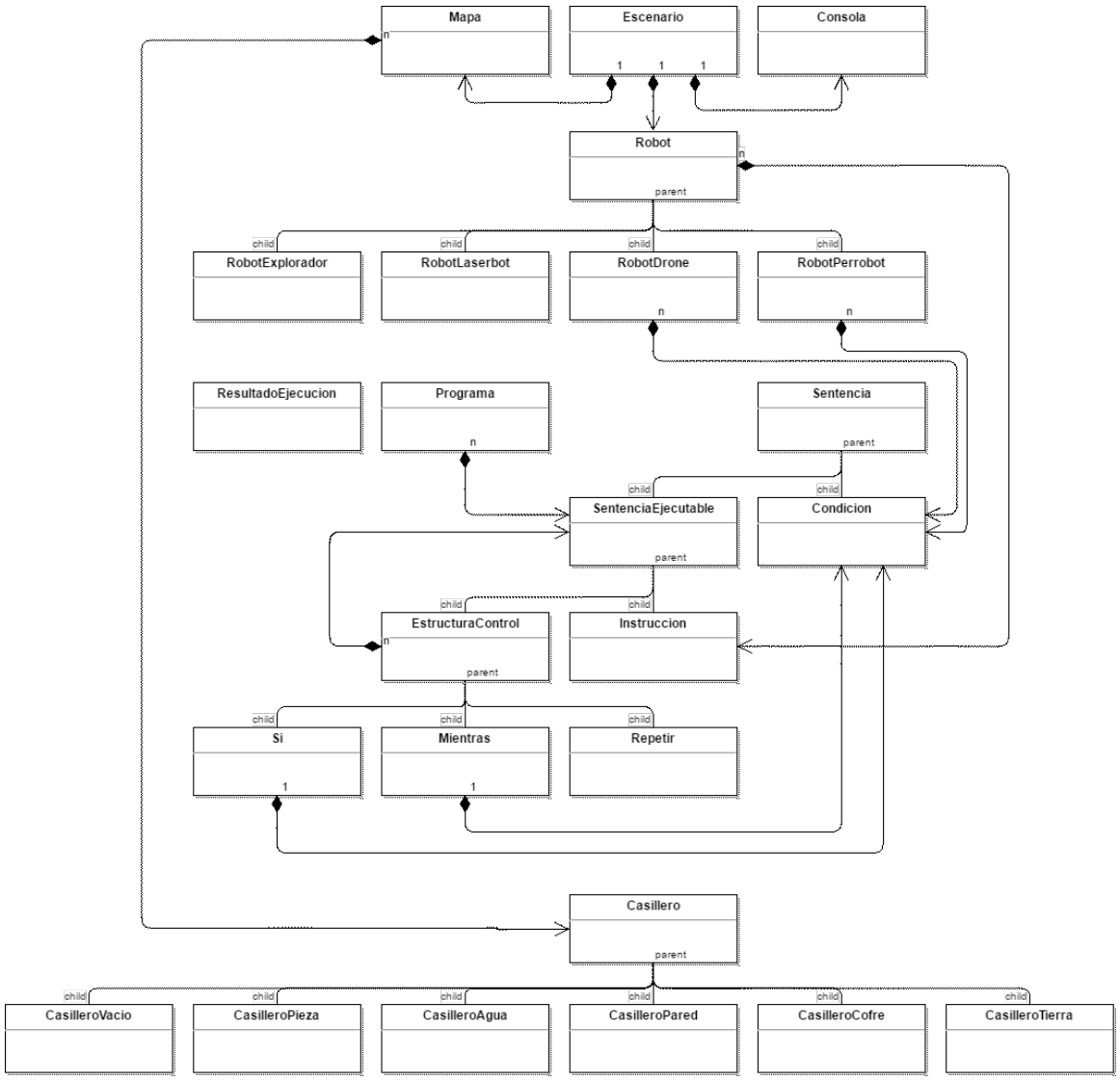

Figura 5.1: Diagrama de clases simplificado de la lógica detrás de los escenarios de programación.

Si bien no parece haber un estándar en la forma de estructurar un proyecto en *Unity*, se optó por una práctica común que es crear directorios para cada tipo de *asset*. En particular, la estructura del proyecto de AstroCódigo tiene la siguiente forma:

- *Animations*: este directorio contiene todos los clips de animaciones y los controladores de animación de todos los modelos 3D y sprites 2D existentes en el proyecto. A su

vez, el contenido de este directorio está organizado en base a las distintas secciones del juego, por ejemplo, escenarios, planetas, cinemáticas, etc.

- *External Assets*: este directorio contiene todos los *assets* desarrollados por terceros, es decir, todos aquellos que fueron descargados desde la *Unity Asset Store*.
- *Fonts*: aquí se encuentran las fuentes de texto utilizadas en el juego.
- *Models*: en este directorio se ubican todos los modelos 3D creados en *Blender* y los terrenos generados en *Unity* para los planetas.
- *Prefabs*: aquí pueden encontrarse todos los *prefabs* del juego. Los *prefabs* son objetos prefabricados que sirven de plantilla a la hora de instanciar nuevos objetos en la escena de juego. Esto permite instanciar objetos en tiempo de ejecución con componentes y propiedades configuradas previamente.
- *Scripts*: aquí se tienen todos los scripts C# que le dan comportamiento y lógica a los distintos objetos del juego. Al igual que el directorio *Animations* y otros, este directorio se encuentra subdividido por las distintas secciones del juego con la intención de ordenar la gran cantidad de *scripts* desarrollados.
- *Sonidos*: aquí se hallan todos los efectos de sonido y pistas musicales utilizadas en el juego.
- *Standard Assets*: este directorio contiene los *assets* estándar de *Unity* que, por cuestiones de eficiencia, no vienen importados por defecto, sino que hay que importarlos explícitamente.
- *Texturas*: aquí se encuentran todas las texturas del juego.
- *Tutoriales*: este directorio es la excepción a la lógica detrás de la estructura del proyecto, ya que no contiene un tipo particular de *asset*, sino más bien todos los *assets*, independientemente de su tipo, relacionados a los tutoriales del juego.

El proceso de desarrollo comenzó por las partes más importantes del juego, siguiendo con las más accesorias. Los escenarios de programación fueron el primer problema a resolver, creando su interfaz, la inicialización del desafío de programación, la mecánica de armado de programa y la ejecución del programa armado. Al mismo tiempo que se desarrollaron los escenarios de programación, se implementó la herramienta web para la

creación de escenarios personalizados (ver sección 5.3.1). Luego se procedió a realizar la conexión con la API y la creación del menú principal del juego, lo cual permitió integrar los escenarios personalizados creados. A continuación, se trabajó en la interfaz y la navegación de la parte del juego en la cual se exploran los planetas. Posteriormente, se desarrolló la mecánica circundante a las misiones secundarias, y por último, se integraron los tres planetas al mismo tiempo que se trabajó sobre las cinemáticas de la historia.

Durante el desarrollo surgieron numerosos desafíos y dificultades. Entre las dificultades que se presentaron se puede destacar la decisión de cambiar la plataforma del juego de WebGL para ejecutar en un navegador, a Windows/Linux/Mac ejecutando como aplicación de escritorio. Este cambio de dirección se dio debido al bajo rendimiento y alto consumo de memoria RAM de WebGL. Incluso al intentar bajar al mínimo la calidad de los *assets*, el rendimiento de dicho motor gráfico estaba lejos de ser el esperado. Además, compilar para dicha plataforma no soporta la utilización de videos como texturas, característica vital para los tutoriales. Este cambio implicó que el juego deba ser descargado desde la web en vez de ser jugador directamente allí. Sin embargo, esta decisión demostró ser acertada ya que al terminar el desarrollo el peso aproximado del juego era de alrededor de 500 MB, lo cual sería demasiado para cargar en un navegador web.

Otra de las dificultades encontradas durante la implementación del juego fue el incremento de uso de RAM de 400 MB a 8 GB al momento de integrar los planetas. Durante días se trabajó en distintas optimizaciones para mejorar el uso de memoria sin resultados significativos. Finalmente, gracias a un estudio detallado de los resultados arrojados por el *Profiler*[1] de *Unity*, se detectó que la mayor parte del uso de memoria provenía de los recálculos de las animaciones en cada *frame*. Este cálculo resultaba especialmente estresante para la computadora debido a la utilización de *Mesh Colliders*[2] en objetos animados. Esta práctica no es recomendada debido al costo de calcular los bordes del *Mesh Collider* en cada frame. Este problema de rendimiento se solucionó al reemplazar los *Mesh Colliders*

[1]Herramienta que permite, entre otras cosas, medir el consumo de memoria y procesamiento de varios aspectos del juego.

[2]Áreas de colisión cuya forma se calcula y se adapta al contorno de un objeto 3D.

por *Box Colliders*[3].

Entre los desafíos encontrados en el desarrollo se puede destacar: la capacidad de poder cargar un escenario con todas sus particularidades en base a un objeto JSON (ver Fig. 5.2), característica necesaria para permitir a los usuarios crear sus propios escenarios; la capacidad de arrastrar y soltar elementos (instrucciones, estructuras de control y condiciones), funcionalidad que debió ser programada desde cero; la capacidad de ejecutar un programa construido, lo cual requirió del uso de múltiples co-rutinas y estructuras de control tipo pila para soportar la ejecución de estructuras de control anidadas; la conexión con la API, cuya dificultad radicó en el uso de la clase *WWW* provista por *Unity*.

```
{
    "id": 1,
    "nombre": "Escenario 1",
    "descripcion": "En este escenario aprenderás ...",
    "deberiaUsarMientras": false,
    "deberiaUsarRepetir": {"explorador": false, "laserbot": false, "drone": false, "perrobot": false},
    "deberiaUsarSi": false,
    "cantidadOptimaInstruccionesEjecutadas": {"explorador": 7, "laserbot": 7, "drone": 7, "perrobot": 7},
    "cantidadOptimaInstruccionesEscritas": {"explorador": 7, "laserbot": 7, "drone": 7, "perrobot": 7},
    "casilleros": [
        {"tipo": "vacio", "x": 1, "y": 1},
        {"tipo": "vacio", "x": 1, "y": 2},
        {"tipo": "vacio", "x": 1, "y": 3},
        {"tipo": "vacio", "x": 2, "y": 1},
        {"tipo": "vacio", "x": 2, "y": 2},
        {"tipo": "vacio", "x": 2, "y": 3},
        {"tipo": "vacio", "x": 3, "y": 1},
        {"tipo": "vacio", "x": 3, "y": 2},
        {"tipo": "pieza", "x": 3, "y": 3}
    ],
    "tutoriales": ["inicial", "pieza", "explorador"],
    "robotsDisponibles": ["explorador", "drone"],
    "estructurasDisponibles": [],
    "esEscenarioCustom": false
}
```

Figura 5.2: JSON que representa al primer escenario del juego.

5.2.1. Tutoriales

El desarrollo de los tutoriales nació como respuesta a una necesidad observada en las pruebas preliminares con usuarios. En estas pruebas se notó que los jugadores tenían dificultades en entender las mecánicas y objetivos del juego. Se decidió así confeccionar tutoriales que explicaran estas cuestiones. Al principio se pensó en utilizar una serie de imágenes

[3]Áreas de colisión de forma cúbica.

que ilustraran el funcionamiento del juego junto a un texto descriptivo acompañando cada imagen. Para comprobar la efectividad de este formato se realizaron nuevamente pruebas con usuarios. En estas pruebas se evidenció el hecho de que los jugadores encontraban estos tutoriales demasiado extensos y tediosos de leer, lo que provocaba que los ignoraran e intentaran directamente jugar. Fue claro que era necesaria una modificación en el formato de los tutoriales por uno más dinámico, por lo que se consideraron dos posibilidades: reemplazar las imágenes por GIFs animados o vídeos. Luego de un análisis comparativo se demostró que los videos resultaban más fluidos, no sumaban un peso considerable sobre los GIFs y ofrecían la posibilidad de sumar audio sobre las animaciones. Así se implementó una solución que utiliza videos que relatan el texto descriptivo, el cual aún está presente a un lado de los mismos. Se realizaron nuevamente pruebas donde este formato fue bien recibido, por lo tanto es el que se presenta en el juego final.

Para confeccionar los videos de los tutoriales se utilizó un software que permite la grabación de pantalla, llamado Camtasia[4]. Con este fue posible capturar ciertas escenas seleccionadas mientras se jugaba el juego, las cuales se consideró claves para explicar las mecánicas y objetivos del mismo.

Una vez definido el formato de los tutoriales, para mantener al jugador lo más inmerso posible en la temática del juego, se decidió distorsionar las voces que relatan los tutoriales para que se asemejara a la de un robot. Para esto se utilizó el programa *Audacity*[5], el cual permite introducir diferentes filtros sobre el audio.

Los tutoriales son presentados al inicio de cada escenario de programación, y cada uno explica los conceptos necesarios para resolver el escenario en cuestión. Es posible que los tutoriales puedan ser revisitados por el jugador desde una opción del menú del juego que permite acceder a todos los tutoriales disponibles.

Es posible que dentro en un escenario en particular, se pretenda entrenar más de un concepto, por lo tanto se necesita mostrar más de un tutorial. Para esta situación se implementó

[4]Software de grabación y edición de video. http://discover.techsmith.com/camtasia-brand-desktop/

[5]Software de código abierto para edición de audio. http://www.audacityteam.org/

un sistema de pasos o *slides* que organiza los distintos tutoriales, poniendo a disposición del usuario opciones para avanzar o retroceder entre los distintos temas o saltearlos por completo.

5.3. API y sitio web

Dado que la información de cada jugador se encuentra en una base de datos remota, es necesaria una constante comunicación a través de internet entre el juego y dicha base de datos. Para lograr tal comunicación, se implementó una API web que provee los puntos de entrada necesarios para la obtención, actualización y creación de los datos almacenados en la base de datos anteriormente mencionada. Los puntos de entrada que expone la API son:

- ***POST** /api/authenticate*: permite a un usuario iniciar sesión con su cuenta.
- ***GET** /api/ping*: permite a un usuario determinar si su comunicación con la API está establecida correctamente.
- ***GET** /api/user*: permite a un usuario autenticado obtener la información completa sobre su usuario, incluyendo su progreso en el juego.
- ***POST** /api/progress*: permite a un usuario autenticado actualizar su progreso en el juego.
- ***GET** /api/customScenarios*: permite a un usuario autenticado obtener todos los escenarios personalizados además de su propio progreso en cada uno de ellos.
- ***POST** /api/get_progress_custom*: permite a un usuario autenticado obtener el progreso de un jugador dado para un escenario personalizado dado.
- ***POST** /api/progressCustom*: permite a un usuario autenticado actualizar su progreso sobre un escenario personalizado dado.

La API fue desarrollada sobre el *framework* de PHP *Laravel* 5.3. La elección de dicho *framework* se dio principalmente debido a su clara estructuración del patrón MVC, su reputación, su documentación y su comunidad activa. Además se hizo uso de la librería

JSON Web Tokens para *Laravel* lo cual permitió la implementación de *tokens* de uso único que brinda una mayor seguridad para los usuarios en los intercambios de información con la API.

La base de datos fue creada utilizando el motor MySQL debido a los conocimientos previos sobre esta tecnología y por el soporte por defecto que brinda *Laravel* sobre este motor. La estructura de la base de datos es bastante simple y se compone solo de tablas correspondientes a usuarios, progresos de los usuarios en el juego, escenarios personalizados y los progresos de los usuarios sobre los escenarios personalizados (ver Fig. 5.3).

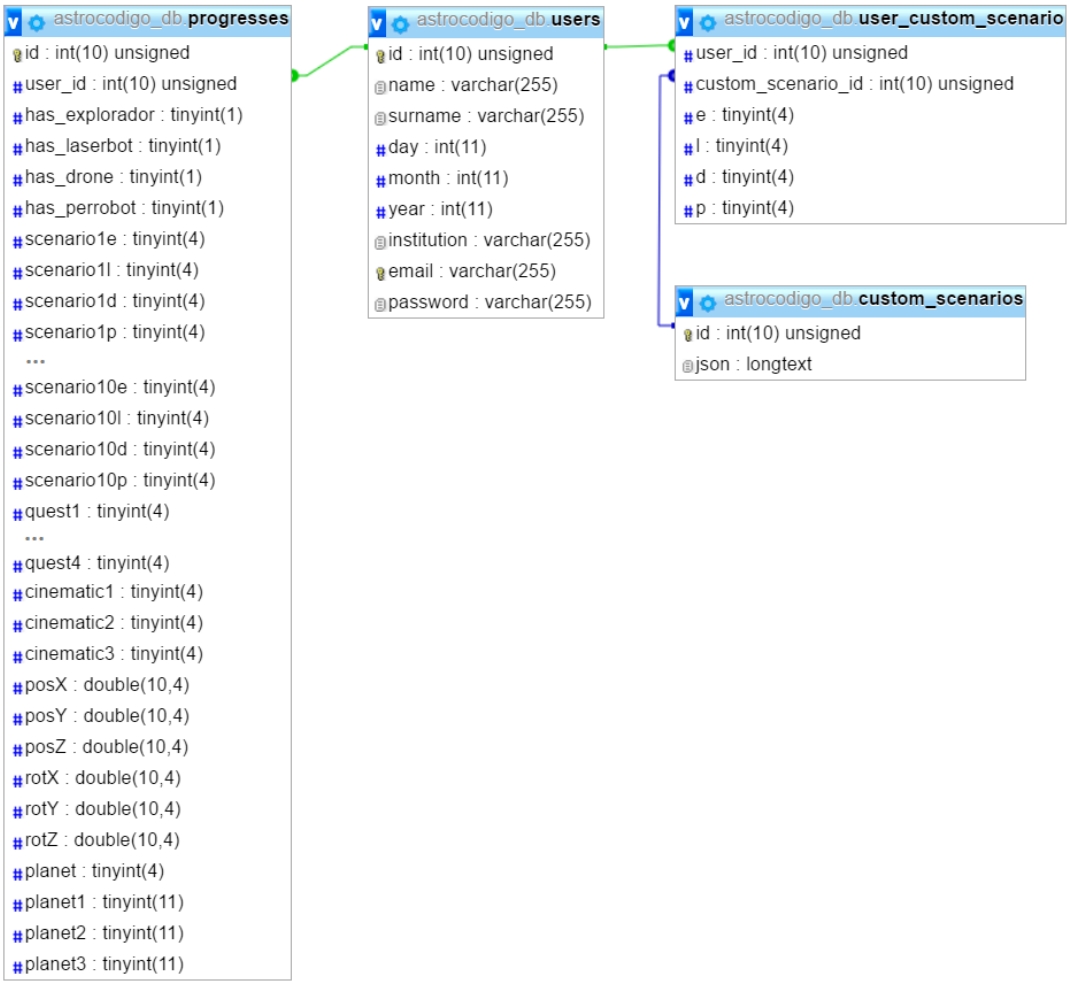

Figura 5.3: Estructura de la base de datos del juego.

Por otro lado, se aprovechó el mismo proyecto en *Laravel* para alojar el sitio web del juego, donde los usuarios pueden registrar sus cuentas, descargar el juego, ver el ranking de jugadores, acceder a la herramienta de creación de escenarios personalizados (ver sección 5.3.1) y ponerse en contacto con los desarrolladores del juego. Para el desarrollo este sitio se utilizaron tecnologías como HTML5, CSS3, Bootstrap y JQuery, con el fin de lograr un diseño adaptable a cualquier dispositivo y una experiencia de usuario agradable.

5.3.1. Generador de escenarios personalizados

Como se explicó en el Capítulo 4, el Generador de Escenarios es una herramienta que permite diseñar y crear escenarios de programación del juego, con el objetivo de agregar desafíos a los ya existentes (Ver sección 4.9).

A lo largo de su desarrollo, el generador tuvo una importante cantidad de revisiones y mejoras. En primera instancia, se desarrolló el generador como una herramienta sencilla formada por celdas en una tabla HTML. Cada celda podía ser pintada de diferentes colores que representaban a los diferentes objetos que puede posicionarse en cada casillero. De esta forma se creaba el escenario en un formato de dos dimensiones y luego se traducía a 3D en el juego. Posteriormente, se sumó la posibilidad de elegir los tutoriales que se mostrarían al iniciar el escenario de programación creado, junto con los robots y estructuras de control que estarían disponibles para resolverlo.

Luego de realizar varias pruebas de usabilidad, se optó por cambiar el mecanismo de creación de escenarios. Se decidió reemplazar el entorno de construcción en 2D por uno en 3D. De esta forma, se brinda una mayor claridad al proceso de construcción ya que la visualización del escenario que se crea en el generador es similar a la del juego. Para llevar a cabo esta tarea se trabajó con la librería web 3D *Babylon JS*, la cual permitió crear un entorno 3D dentro del sitio web. En este entorno se pueden colocar los casilleros dentro del escenario, rotar el escenario 360 grados y visualizarlo de una manera muy cercana a cómo se verá luego en el juego (ver Fig. 4.22).

La librería *Babylon JS* es un motor gráfico 3D de código abierto que hace uso de WebGL, una API en *JavaScript* que permite generar gráficos 2D y 3D interactivos dentro de un navegador web compatible sin la necesidad del uso de *plugins*, y *JavaScript* como lenguaje base para acceder a sus funciones de programación. Se eligió a *Babylon JS* por sobre otras librerías similares para el diseño web 3D, pues poseía herramientas y ejemplos que se ajustaban perfectamente a las necesidades de construir un generador de escenarios como el deseado. Además, cuenta con una herramienta online que permite que los códigos sean ejecutados y sus resultados vistos en vivo llamada *Babylon Playground*, la cual probó ser una forma muy útil para la observación y testeo de nuevas funcionalidades. Los gráficos generados por *Babylon* se construyen principalmente por código. Mediante funciones propias del motor se crean los modelos 3D en un lienzo y se posicionan indicando coordenadas en los 3 ejes cartesianos. Aún así, existe la posibilidad de importar modelos más complejos de otros programas de diseño, como *Blender*, pero en este caso, esto no fue necesario.

Tal como se mencionó en el Capítulo 4, una funcionalidad importante del generador es mostrar al usuario la forma óptima de resolver cada escenario en el juego. Esto implica calcular el camino mínimo para cada robot y, a partir de este, construir el algoritmo que se deberá crear dentro del juego. La implementación de esta funcionalidad resultó especialmente desafiante y se describe a continuación.

Para calcular el camino mínimo de cada uno de los robots, se utilizó el algoritmo de Dijkstra, tomando la grilla del escenario como un grafo, y asignando pesos muy elevados a las posiciones que tuvieran obstáculos que el robot no pudiera sortear. Se consiguió un algoritmo que devuelve el mejor camino para alcanzar cada uno de las piezas presentes en el escenario. Para lograrlo, fue necesario realizar modificaciones sobre el algoritmo Dijkstra original para adaptarlo a las particularidades de este diseño de juego. Una de las complicaciones que surgieron en este punto es el hecho de que Dijkstra no tiene en cuenta las direcciones de movimiento en un grafo. En este caso, es necesario tener en cuenta esto último ya que la dirección afecta el peso del siguiente movimiento del robot. Para orientarse en una dirección se debe realizar un giro, lo que significa un movimiento extra, lo cual afecta al peso final del recorrido. Por ejemplo: si el robot está en dirección Norte, continuar hacia

el Norte no suma nada al peso original de la arista. Girar hacia el Este u Oeste significa hacer un giro a derecha o izquierda respectivamente, lo cual insume un movimiento extra al peso de la arista. Y en caso de tener que dirigirse hacia el Sur, significa realizar dos giros a la izquierda o dos giros a la derecha, lo que insume dos movimientos extra.

El algoritmo creado para la búsqueda del camino mínimo en un escenario se utiliza tanto para mostrar gráficamente el camino que deberá recorrer cada robot, como para generar la secuencia de instrucciones de la solución óptima al escenario. Para la creación de la secuencia óptima fue necesario procesar la secuencia de instrucciones obtenida del algoritmo en busca de patrones de instrucciones contiguas que deberían estar contenidas en un *repetir*. La implementación del algoritmo que permitiera tal procesamiento requirió iterar la secuencia de instrucciones buscando patrones de instrucciones del mayor tamaño posible (n/2 instrucciones, siendo n el tamaño de la secuencia completa) en la primer iteración, e ir decrementando hasta patrones de tamaño 1. La creación de este algoritmo presentó numerosos desafíos y actualmente se encuentra en una versión inicial de prueba habiendo arrojado en la mayoría de los casos resultado válidos. Actualmente se trabaja en la mejora del mismo para los casos especiales donde no se arribó a la solución esperada.

Habiendo avanzado en el desarrollo del generador se realizaron nuevas pruebas y revisiones. Se notó que la complejidad del proceso de creación provocaba que fuera poco intuitivo. Se optó por cambiar el proceso y disponer las herramientas en un sistema guiado por pasos (también conocido como *wizard*), de tal manera que el usuario pueda completar los distintos requerimientos de un escenario de una manera más ordenada y coherente.

El wizard de construcción del escenario está compuesto de cuatro pasos:

1. El primer paso es una configuración inicial sencilla que permite indicar el nombre, descripción y tamaño del escenario que se construirá.

2. En el segundo paso, se encuentra el entorno 3D para construir el escenario, y por debajo de este se muestra una serie de botones correspondientes a los distintos tipos de casillero (ver Fig. 4.22).

3. El tercer paso se destinó para la selección de los tutoriales a mostrar cuando se inicia el escenario en el juego, junto con los robots y estructuras de control que estarán disponibles.

4. El cuarto y último paso contiene varias funcionalidades. Primero se muestra un pequeño entorno 3D, de la misma manera que en el paso dos, pero sin las funcionalidades de construcción. Este entorno se utiliza para mostrar gráficamente el mejor recorrido para recoger todas las piezas, según el robot que se seleccione de una lista. También se puede visualizar, en la parte de la derecha, el programa construido siguiendo el camino mínimo según las estructuras que se hayan marcado como disponibles en el tercer paso. Y por último, presionando en el botón "Guardar escenario", el escenario construido se guardará en la base de datos, siempre y cuando no se encuentre un error crítico de construcción del escenario, en cuyo caso se muestra un cartel notificando el error (ver Fig. 5.4).

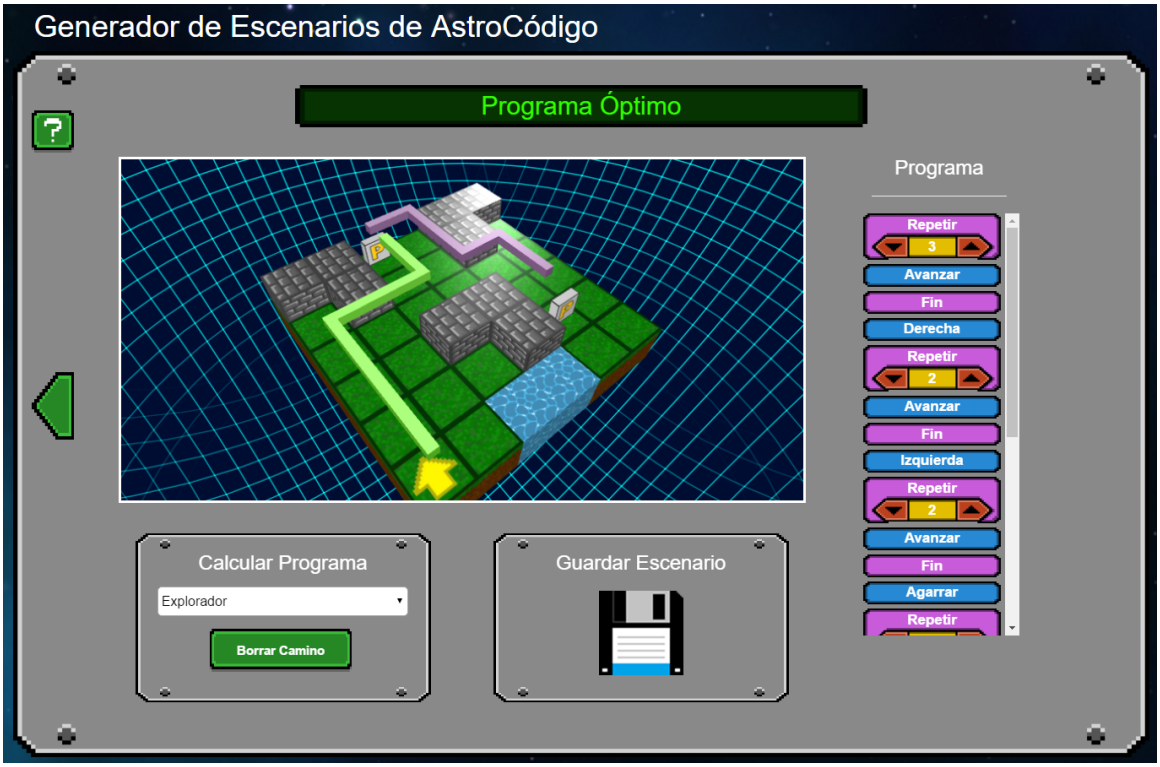

Figura 5.4: Generador de escenarios - Programa Óptimo.

La consola de errores se desarrolló con el objetivo de mostrar mensajes importantes durante el proceso de construcción (ver Fig. 4.22). La comprobación de errores existe mayormente para evitar que escenarios irresolubles lleguen a estar disponibles en el juego, por lo tanto se muestran errores críticos cuando, por ejemplo, no hay ninguna pieza en un escenario, o un robot es incapaz por sí solo de recoger todas las piezas para resolverlo. Esta consola se construyó de una manera similar a la del juego, con los mensajes coloreados de la misma forma, ya que se pretendía que tanto el juego como el Generador de escenarios coincidieran en estética, dando a entender que ambas son partes de una misma experiencia.

5.4. Diseño e implementación de modelos y figuras

En esta sección se presentan los distintos procesos y herramientas que se utilizaron, así como también los inconvenientes que surgieron, al crear los modelos y figuras que se utilizaron en el juego. Para esta presentación los modelos y figuras se agruparán de acuerdo a su tipo: 2D, tales como las interfaces y menús; y 3D, tales como los robots, bloques de escenarios de programación, y mapas que se recorren en primera persona.

5.4.1. Modelos 2D

Para la construcción de los menús y las interfaces del juego, ya sean los distintos paneles, botones, inputs, diálogos y demás, se utilizaron las herramientas que provee *Unity* para la creación de interfaces, aprovechando el sistema de anclaje que permite que la interfaz se adapte a cualquier resolución y tamaño de pantalla. La mayor parte de los componentes de interfaz están contenidos en paneles que asemejan pantallas antiguas dentro de consolas metálicas con tornillos, esto, sumando al estilo *pixel art*, le da al juego una sensación de tecnología *vintage*. Para el diseño e ilustración de estos componentes se hizo uso del programa *Piskel*, el cual está orientado a la realización de *pixel art*, es decir, que permite ilustrar pixel por pixel e incluso facilita la creación de animaciones simples,

característica que también fue aprovechada en el desarrollo de AstroCódigo. Se buscó lograr una consistencia estética entre el sitio web, el generador de escenarios y el juego, por lo que muchos de los diseños 2D (botones y paneles) fueron reutilizados en varias ocasiones. De esta forma se intenta mostrar que las tres herramientas forman parte de un todo y trabajan juntas para enriquecer la experiencia del usuario.

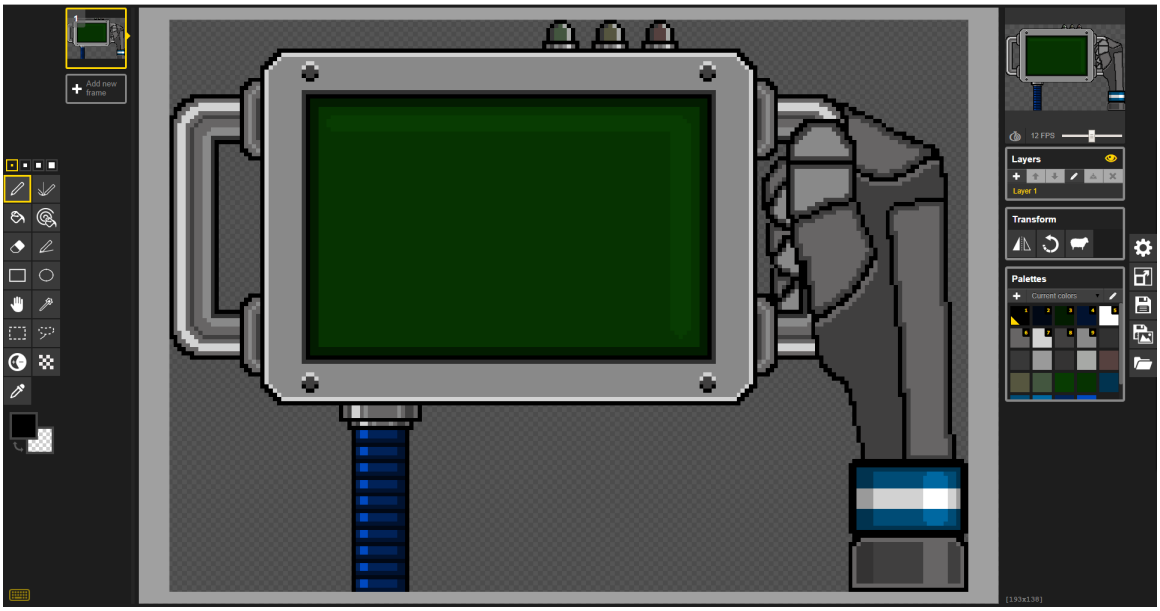

Figura 5.5: Diseño de la consola de los escenarios de programación en Piskel.

5.4.2. Modelos 3D

Para crear los modelos 3D que se utilizan en el juego se optó por la herramienta de modelado 3D *Blender*, ya que es una herramienta libre y con una gran comunidad de desarrollo y ejemplos de donde obtener información. *Blender* es un programa multiplataforma dedicado especialmente al modelado, iluminación, renderizado, animación y creación de gráficos tridimensionales. Disponer de una herramienta libre fue muy beneficioso, ya que se contaba con todas sus funcionalidades sin las limitaciones que pueden encontrarse en otros programas, debido a la necesidad del pago de una licencia.

Dado que la estética que se definió para el juego es de *pixel art* se decidió que, para

concordar, la estética de los modelos 3D fuera en su mayoría de un aspecto de baja cantidad de polígonos o *low-poly*. Esto facilitó en gran medida el diseño de los modelos 3D ya que este tipo de estética no requiere gran nivel de detalle (ver Fig. 5.6).

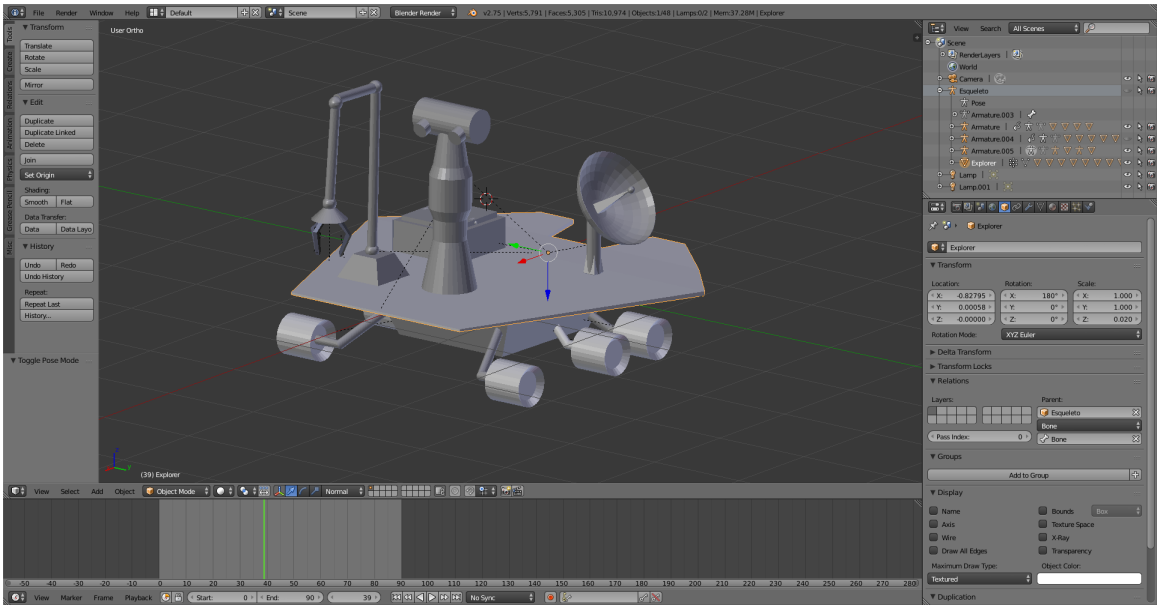

Figura 5.6: Diseño 3D del robot explorador en Blender.

Blender es una herramienta muy potente y la curva de aprendizaje resulta muy empinada. Sin embargo resultó ser una herramienta extremadamente útil para diseñar los modelos 3D necesarios para el juego, especialmente aquellos más complejos que no pueden ser creados en *Unity*. Por ejemplo, el castillo creado para el último Planeta o los robots. Estos últimos al ser animados, debían estar compuestos de varias partes y esto no resultaba posible en *Unity*.

En un principio se utilizó *Blender* para realizar las animaciones que se habían pensado para los modelos 3D, como son las acciones de *excavar* del Perrobot o *destruir* del Laserbot. Pero a medida que se avanzaba con el desarrollo del juego se descubrió que importar los modelos 3D desde *Blender* a *Unity* y animarlos en este último era más sencillo que realizar las animaciones en *Blender* e importarlas luego a *Unity*. Además, las animaciones que se creaban y mostraban correctamente en *Blender*, al ser importadas en *Unity* se mostraban

fragmentadas en diferentes partes y sumaban una complicación al necesitar ser combinadas nuevamente. Las animaciones que se pretendía construir eran simples, y la complejidad de *Blender* presentó un obstáculo. Así fue que para crear estas animaciones se utilizó *Unity* que resultó ser más sencillo y por lo tanto más rápido.

Otro de los desafíos que se encontró en la utilización de *Blender* fue el mapa de texturas de los modelos 3D. En un principio se utilizaron otros programas de edición gráfica para pintar las imágenes que luego se aplican sobre los modelos, pero avanzando en el aprendizaje de *Blender*, se descubrió que dicho programa cuenta con una herramienta que permite pintar sobre los mismo modelos 3D directamente allí. Esto último fue una gran ventaja, ya que permitió ver en el momento cómo quedaban los distintos diseños de los modelos 3D agilizando considerablemente el proceso de texturización.

Para la creación de los terrenos de los planetas, se optó por la utilización de *Unity* ya que ofrece un conjunto de herramientas que están dirigidas específicamente al modelado de terrenos. Con éstas fue posible crear distintas zonas para dotar al juego de variedad en sus niveles. Debido a que el modelado de terrenos utiliza mapas de calor o *heatmaps*, los cuales sólo permiten moldear elevaciones o hendiduras en los terrenos, no fue posible modelar las cuevas que se encuentran en los distintos planetas, por lo tanto se recurrió nuevamente a *Blender* para crear estas formaciones.

Para acelerar el proceso de desarrollo y dotar de variedad a los mapas dentro del juego, se hizo uso de modelos 3D adquiridos desde la *Unity Asset Store*. Algunos de estos modelos son los árboles, las rocas, la lava y las lámparas, también la cápsula de salvamento en la que el astronauta es eyectado de su nave y las naves tanto del astronauta como de los piratas espaciales.

5.5. Conclusiones del capítulo

En este capítulo se realizó una descripción de la etapa de desarrollo de AstroCódigo. Se detallaron las herramientas utilizadas, las experiencias adquiridas y los inconvenientes que surgieron. Lograr implementar todos los componentes del juego diseñados en el capítulo anterior implicó el uso de numerosas herramientas que en algunos casos requirieron capacitación para su uso por parte de los tesistas. Durante esta etapa se realizaron pruebas informales que permitieron realizar validaciones sobre las decisiones tomadas en relación al juego y ajustar los aspectos necesarios. Para las pruebas informales se invitó a diferentes personas a utilizar las funcionalidades disponibles, entre ellos también a los directores del trabajo.

En el siguiente capítulo se describen las sesiones de prueba realizadas con AstroCódigo y los resultados alcanzados.

CAPÍTULO 6

Sesiones de prueba con AstroCódigo

6.1. Introducción

En el presente capítulo se detalla la etapa de evaluación de AstroCódigo para validar que los desarrollos realizados y explicados en este documento cumplan con los objetivos propuestos y conocer las opiniones de los usuarios. Para esta evaluación se realizó una prueba con una muestra del grupo destinatario compuesto por jóvenes en edades de educación secundaria o comienzos de educación universitaria. La prueba consistió en una sesión de juego seguida por una encuesta. Esta encuesta estaba conformada por 2 partes. La primera parte se enfocó en la experiencia del juego y la segunda parte en la motivación del jugador. Para esta segunda parte se utilizó el instrumento IMI (*Intrinsic Motivation Inventory*). A continuación se detalla todo el proceso y sus resultados.

6.2. Sesiones de pruebas informales de AstroCódigo durante el desarrollo

Como se ha mencionado en capítulos anteriores, en diferentes momentos del proceso de desarrollo del sistema presentado en este trabajo se realizaron pruebas informales. Las pruebas son denominadas informales debido a que no establecen tiempos, ni perfiles específicos de usuario. Estas pruebas se realizaron repetidas veces, durante aproximadamente 3 meses con usuarios cercanos al contexto de los desarrolladores y con los directores. Permitieron evolucionar las 3 partes del sistema: el juego, el generador y el sitio web a partir de las opiniones de los participantes. Las pruebas tuvieron como objetivo encontrar errores en cada caso y analizar la interacción del jugador. Con estas pruebas se detectaron cuestiones relacionadas a la interfaz y a la jugabilidad que fueron mejoradas, entre las cuales se destacan:

- el agregado de paneles explicativos de la función de cada componente, los cuales se muestran al deslizar el mouse sobre el componente en cuestión;
- el agregado de un camino dentro del modo en primera persona que muestre cómo llegar a los escenarios y misiones secundarias, ya que algunos eran difíciles de encontrar;
- la modificación de la leyenda de la instrucción *Volver* por *Terminar*, ya que no expresaba correctamente su función y confundía a los jugadores;
- el agregado de una explicación sobre la interfaz de movimiento del personaje en primera persona, ya que no era claro para todos los jugadores;
- la modificación de los colores de los bloques de las estructuras de control para facilitar su identificación en el programa;
- el agregado del botón *Detener*, para detener la ejecución de un programa.

En el caso del diseño de los tutoriales, los participantes expresaron que eran tediosos y extensos y por lo tanto no les prestaban atención, esto originó que se refinaran agregando

video y audio y reduciendo su longitud llegando a un diseño aceptado por los jugadores. Además las pruebas sirvieron para detectar errores de implementación que fueron solucionándose en cada caso. Durante las sesiones se les pedía a los usuarios que expresen su opinión a medida que iban jugando utilizando la técnica *thinking aloud*, reconocida en el ámbito de los test de usabilidad.

6.3. Sesión de prueba formal de AstroCódigo con alumnos ingresantes

Habiendo llegado a una versión estable del sistema se planeó una sesión formal de prueba. Esta prueba tuvo como objetivo evaluar el juego. Se decidió realizar un recorte del juego completo para que la prueba no resultara excesivamente larga. Así, de los tres Planetas que contiene AstroCódigo sólo se incluyó el primero (Capítulo 4, sección 4.6.2). En este planeta, se seleccionaron y distribuyeron accesos a los seis escenarios que se consideraron esenciales para introducir los conceptos de programación que el juego propone desarrollar.

La prueba formal del juego se realizó el día 18-11-2016 en la sala de PC de postgrado de la Facultad de Informática de la UNLP. La muestra utilizada para la prueba estuvo conformada por 10 personas: 7 alumnos aspirantes a las carreras de la Facultad de Informática de la UNLP, un alumno avanzado de la carrera de Licenciatura en Sistemas de la UNLP, 1 alumno de Profesorado de Educación Física y 1 alumno de Diseño de Indumentaria. Los participantes se encontraban principalmente entre los 17 y 20 años, y eran predominantemente de sexo masculino.

Para recoger información de la experiencia se utilizó la técnica de observación participante. Además se realizó un registro fílmico, desde distintos ángulos, para el posterior análisis de las dificultades encontradas por los participantes y las mejoras a realizar en el juego.

La sesión de prueba se estructuró en tres etapas:

1. Presentación del juego y una introducción de los objetivos que se proponía con la sesión.

2. Participación en el juego.

3. Recogida de datos a través de una encuesta y entrevistas informales.

Como ya se mencionó, la prueba fue realizada en una sala de PC, por lo que al llegar cada participante tuvo una computadora disponible con AstroCódigo instalado listo para usar. Luego de la presentación, se entregó a cada participante un nombre de usuario y contraseña aleatorios para poder acceder al juego ya que, como se explicó antes, este cuenta con un sistema de cuentas personales para mantener el progreso de cada jugador.

La prueba tuvo una duración de 45 minutos. El 80 % de los participantes terminaron el juego completo, resolviendo los seis escenarios de programación distribuidos en el juego. Esto será retomado luego, en la sección de Resultados.

6.4. Encuesta

Luego de finalizar el juego, se pidió a cada participante que complete una breve encuesta respecto del juego. La encuesta fue realizada en Google Forms ya que presenta grandes facilidades para la distribución y recolección de los resultados.

La encuesta se dividió en 2 partes. La primera consistió en una serie de preguntas desarrolladas para extraer información sobre el desempeño de los participantes en el juego y sus opiniones sobre el mismo. Y la segunda parte es una serie de preguntas estandarizadas basadas en el cuestionario IMI y que apuntaron a extraer información respecto a la motivación, experiencias y sentimientos de los participantes mientras jugaban.

A continuación se describen ambas partes y sus resultados.

6.4.1. Primera Parte de la Encuesta

La primera parte de la encuesta reveló que el 80 % de los participantes estaban familiarizados con los conceptos de programación presentados en el juego. El 20 % restante respondió que no tenía ningún tipo de conocimiento previo sobre los conceptos presentados, ya que eran los participantes que no pertenecían a la facultad (ver Fig. 6.1).

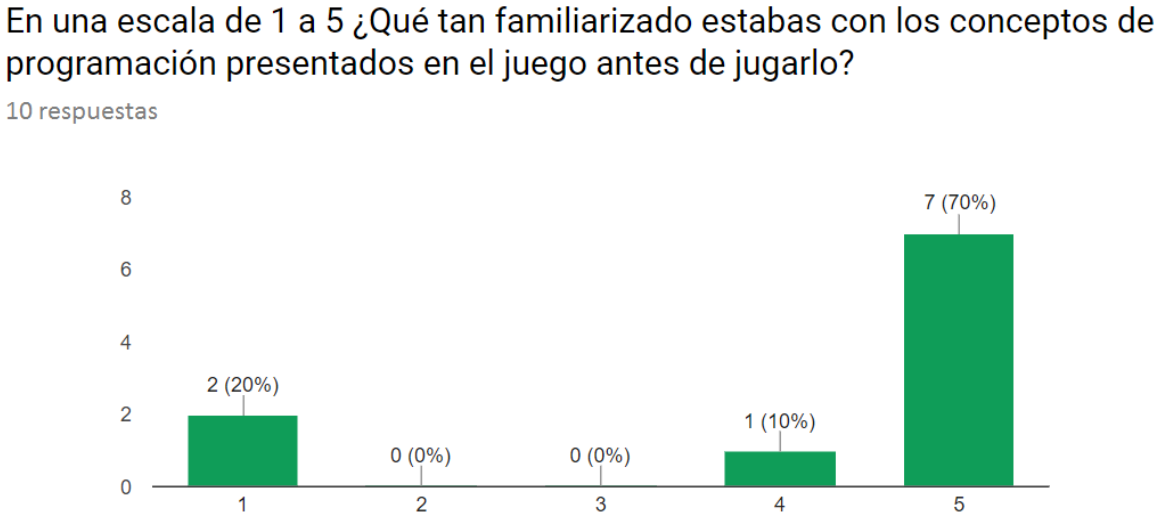

Figura 6.1: Grado de familiaridad con los conceptos de programación de los participante.

Aún teniendo en cuenta lo anterior, 8 de los 10 participantes resolvieron todos los escenarios, y los restantes 2 participantes resolvieron 4 y 5 de los 6 escenarios, por ende se puede deducir que la dinámica del juego fue comprendida y que los tutoriales y guías resultaron útiles para transmitir las mecánicas y objetivos (ver Fig. 6.2).

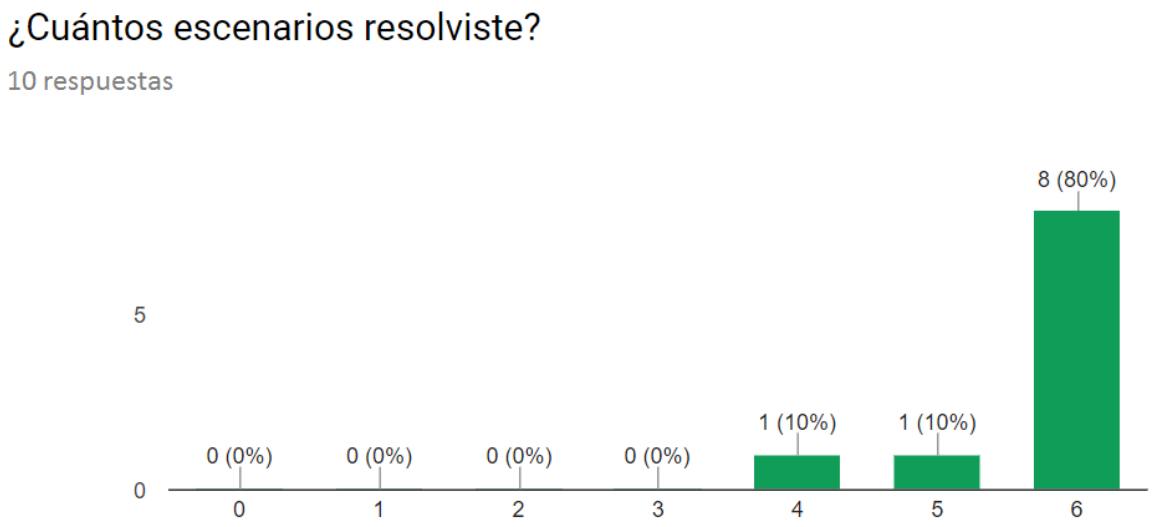

Figura 6.2: Cantidad de escenarios resueltos por los participantes.

Como se explicó en la sección de desarrollo del juego, este también cuenta con misiones secundarias, las cuales son entregadas por personajes extraterrestres y transmiten conceptos teóricos relacionados a la programación (Capítulo 4, Sección 4.7). Cuando se les preguntó a los participantes cuántas de estas misiones pudieron resolver, se obtuvieron resultados similares a los de los escenarios. El 80 % de los participantes resolvió las dos misiones y el otro 20 % resolvió una (ver Fig. 6.3).

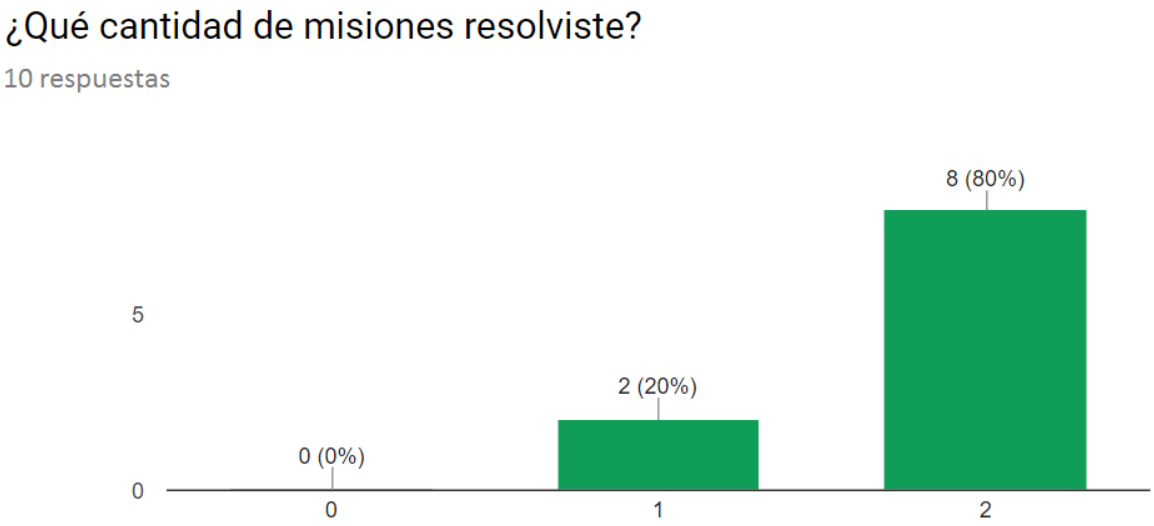

Figura 6.3: Cantidad de misiones resueltas por cada jugador.

Cabe destacar que por más que la cantidad de participantes que resuelve todos los desafíos se repite, no son siempre los mismos los que resuelven todo lo pedido. Por ejemplo, es interesante observar que los 2 participantes externos al curso de ingreso de la facultad de informática, aunque no contaban con ningún tipo de conocimiento previo de programación, resolvieron todos los escenarios y todos los desafíos del juego.

Luego, se pidió a los encuestados que clasifiquen la dificultad del juego asignándole un puntaje en una escala de 1 a 5, donde 1 era muy fácil y 5 muy difícil. Dados que el juego está apuntado a introducir los conceptos de programación en jóvenes antes de que tengan contacto con la informática y que el recorte del juego que se presentó correspondió al nivel inicial del mismo, los resultados fueron los esperados. Un 40 % de los encuestados le dio al juego un puntaje de 2, lo que se traduciría a un "fácil" mientras que el 60 % restante le dio un puntaje de 3, lo cual se traduciría en una "dificultad media". Por lo tanto se puede extraer que para jóvenes de la edad de los encuestados el juego presenta una dificultad moderada (ver Fig. 6.4).

A esta pregunta se le puede sumar la información adquirida de los puntajes obtenidos por los participantes, la cual puede visualizarse en la Tabla 6.1, donde se utilizaron las letras de la A a la J para representar a los 10 participantes de esta sesión de pruebas.

A	B	C	D	E	F	G	H	I	J
6500	4400	6000	7000	5300	7500	3200	2900	3600	4500

Tabla 6.1: Puntaje obtenido por cada participante.

Como se puede observar en la Tabla 6.1, del puntaje máximo que es posible obtener de 9000, el promedio de puntajes resultó más cercano a los 5000 puntos. Estos resultados nos sirven para deducir que, por más que la mayoría de los encuestados completó el juego con facilidad, no se obtuvieron puntajes consistentemente altos, lo cual sugiere que no todos los participantes usaron apropiadamente los conceptos de programación presentados en los tutoriales y que el juego tiene el potencial de poder reflexionar en forma posterior sobre qué

se podría hacer para mejorar estos puntajes.

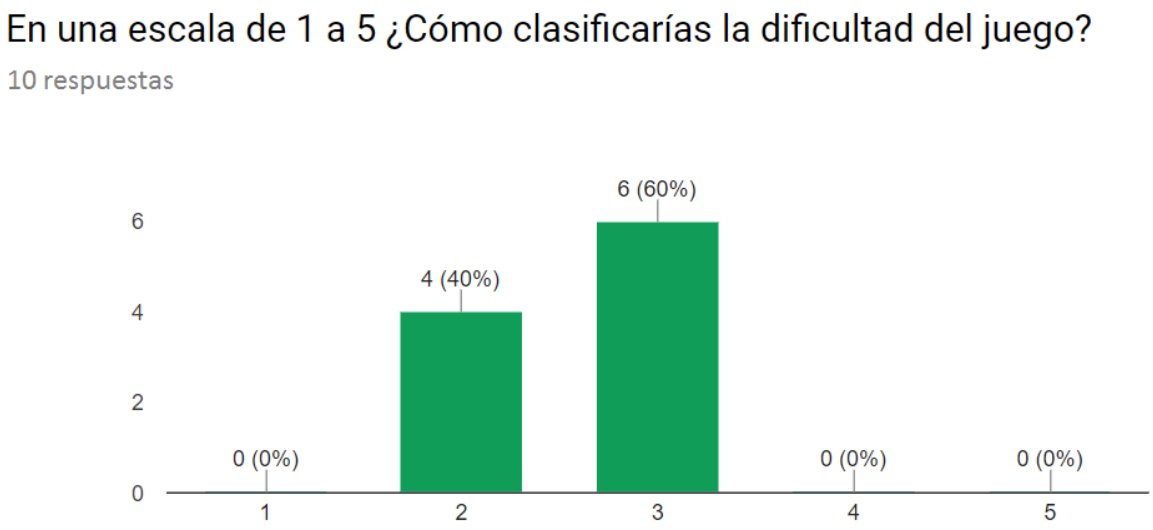

Figura 6.4: Clasificación de la dificultad del juego según los participantes.

Otra pregunta consiste en clasificar la utilidad de los videotutoriales. Estos son lo primero que se presenta al ingresar en los distintos escenarios de programación y tienen como objetivos explicar los objetivos, conceptos de programación y mecánicas del juego (Capítulo 4, Sección 5.4.2). Al igual que en el caso anterior, se pidió que la respuesta fuera un número del 1 al 5, siendo 1 “totalmente inútil” y 5 representa “muy útil”. Las respuestas se presentaron variadas en este caso (ver Fig. 6.5).

Un encuestado respondió con un 2 lo que se traduciría en el hecho de que los tutoriales le resultaron “poco útiles”. Dos encuestado respondieron con un 3 lo que se traduciría en el hecho de que los tutoriales les resultaron “medianamente útiles”. 4 encuestados respondieron con un 4 lo que se traduciría en el hecho de que los tutoriales les resultaron “bastante útiles”. Y los 2 encuestados restantes respondieron con un 5, lo que se traduciría en el hecho de que los tutoriales les resultaron “muy útiles”.

De esta pregunta se puede deducir que los tutoriales, aunque cumplieron su objetivo, son un punto sobre el que se debe seguir indagando, para así analizar si se requieren mejoras y qué tipos de mejoras.

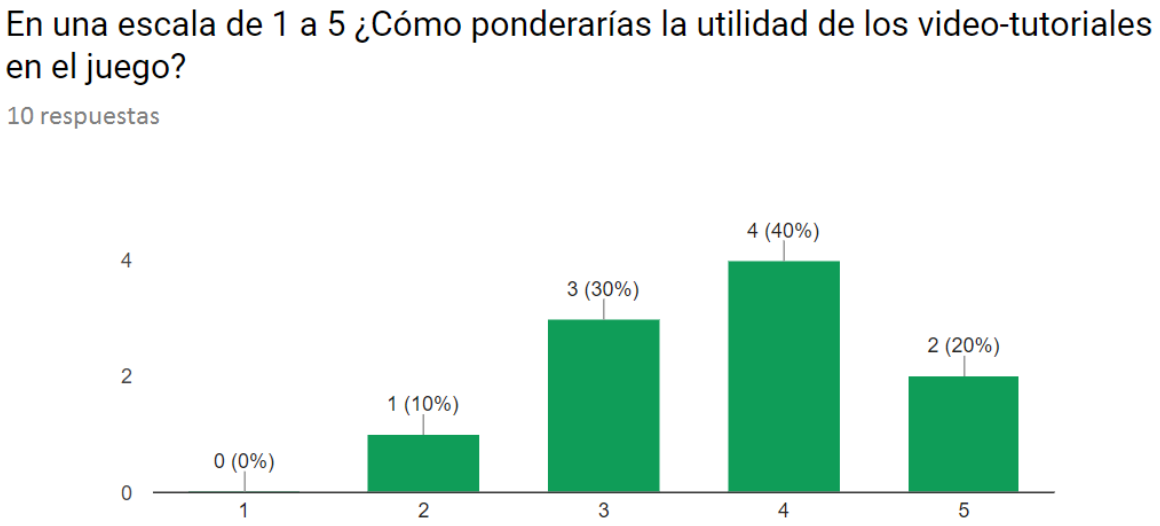

Figura 6.5: Utilidad de los video-tutoriales según los participantes.

La siguiente pregunta pedía clasificar, de la misma manera que las preguntas anteriores, qué tan intuitiva se percibía la interfaz del juego. Tres de los participantes respondieron con un 3, lo que significa que la interfaz les pareció "medianamente intuitiva". Tres de los participantes respondieron con un 4, lo que significa que la interfaz les pareció "bastante intuitiva". Y por último, cuatro de los participantes respondieron con un 5, ya que la interfaz les pareció "muy intuitiva" (ver Fig. 6.6).

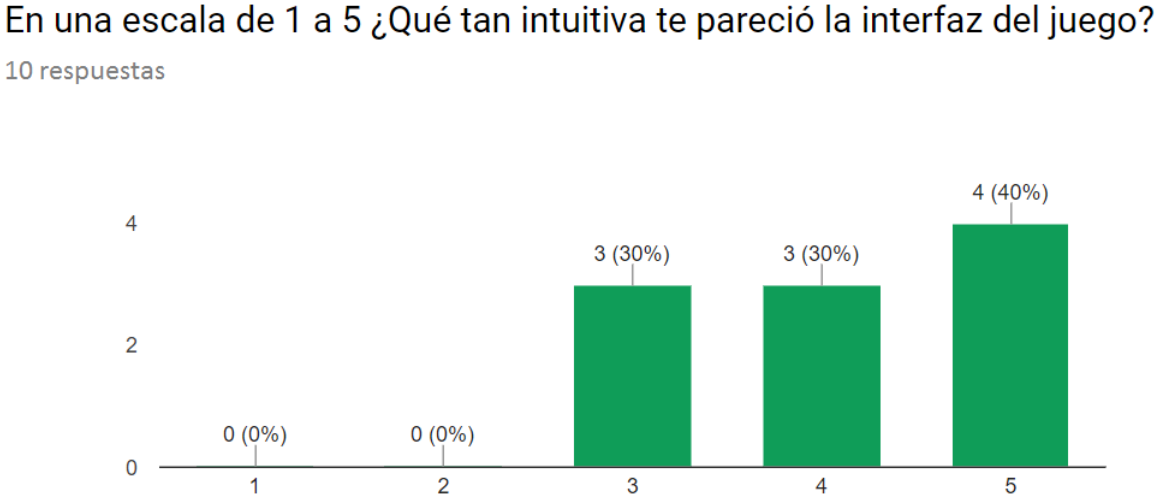

Figura 6.6: Intuitividad de la interfaz según los participantes.

Seguidamente se les pidió a los participantes que catalogaran, según una escala similar a la de las preguntas anteriores, la utilidad del juego a nivel educativo. En este caso el 90 % de los encuestados respondió con un 5 lo que se traduciría en un “muy útil” y solo un 10 % respondió con un 4 lo que significa que le pareció “bastante útil” (ver Fig. 6.7). De las respuestas de los encuestados a esta pregunta se puede deducir que el juego se presenta como una herramienta muy útil a nivel educativo.

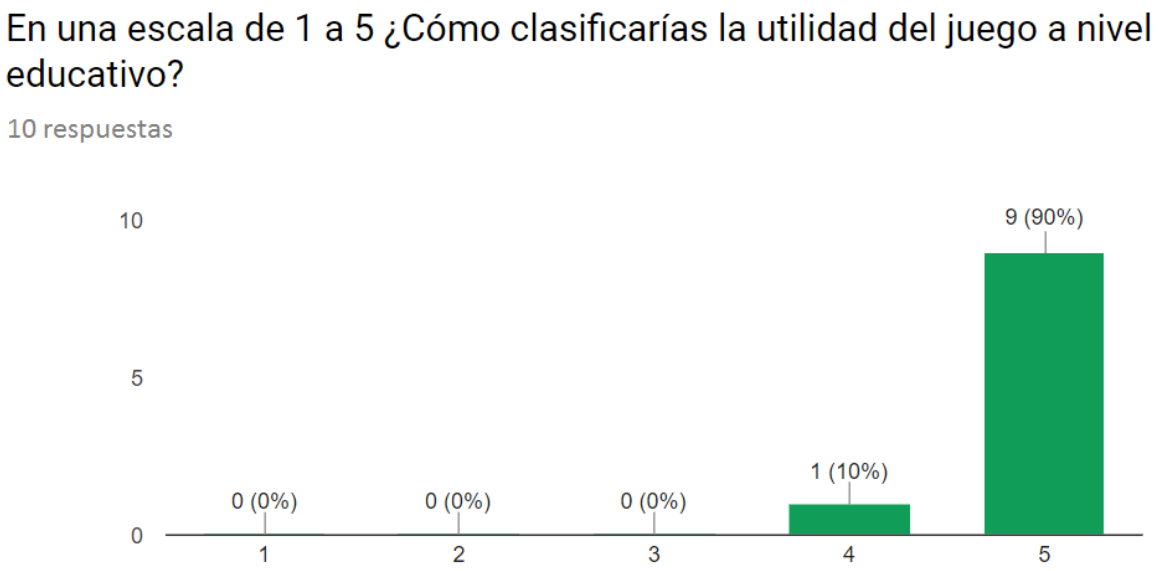

Figura 6.7: Utilidad del juego a nivel educativo según los participantes.

Luego se consultó en qué grado los participantes sintieron que no sabían cómo proseguir con el juego. Se les pidió clasificar sus experiencias en una escala de 1 a 5, siendo 1 “ninguna dificultad para continuar jugando” y 5 “gran dificultad para continuar con el juego”. En este caso 30 % de los encuestados dio un puntaje de 1, lo que se traduce como “ninguna dificultad para continuar jugando”. Un 60 % de los encuestados dio un puntaje de 2, lo que se traduce como “poca dificultad para continuar jugando”. Y el 10 % restante dio un puntaje de 4, lo que se traduce como “bastante dificultad para continuar jugando” (ver Fig. 6.8).

De esto se puede extraer que en general el juego es simple e intuitivo, pero para algunos usuarios requiere mayor acompañamiento inicial. También aquí se debe profundizar para conocer en qué términos les resultaba difícil continuar (resolución de escenarios de

programación o navegación en el juego).

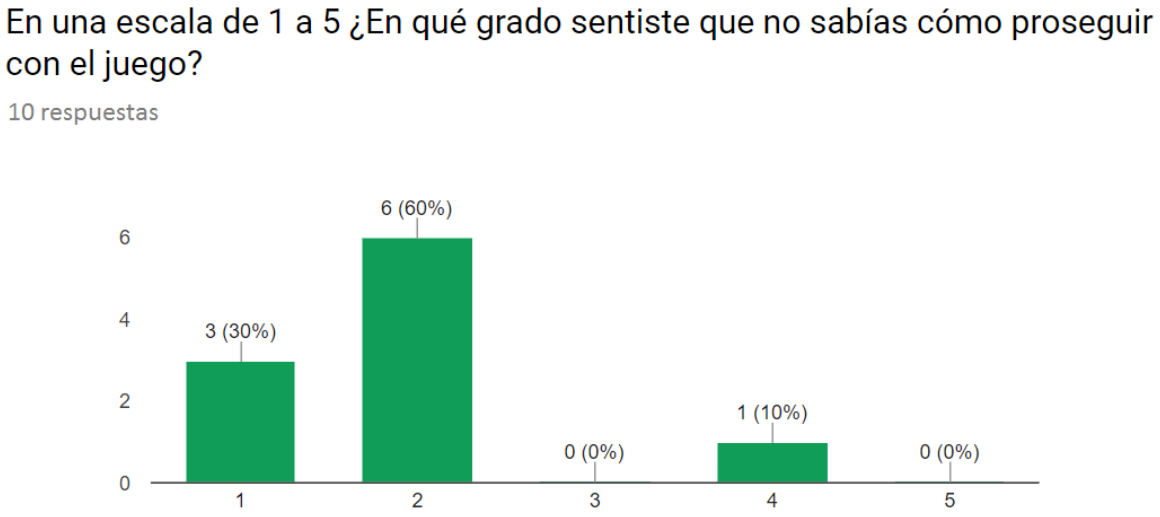

Figura 6.8: Grado en que no se supo cómo proseguir en el juego según los participantes.

La siguiente pregunta requería indicar en qué grado les parecía que el juego podía motivar a un joven a estudiar carreras afines a la informática. De la misma forma que las preguntas anteriores, se utilizó un escala de 1 a 5, siendo 1 el equivalente a "nada motivador" y 5 el equivalente a "muy motivador". En este caso, un 40 % de los encuestados respondió con un 5, indicando que les parecía que el juego podía llegar a ser "muy motivador" para que un joven elija una carrera de informática. Un 40 % respondió con un 4, lo que se traduciría a un "bastante motivador" y el 20 % restante respondió con un 3, lo que se traduciría como "algo motivador" (ver Fig. 6.9). En base a estas respuestas, se vislumbra que el juego es una herramienta con un potencial de motivación alto.

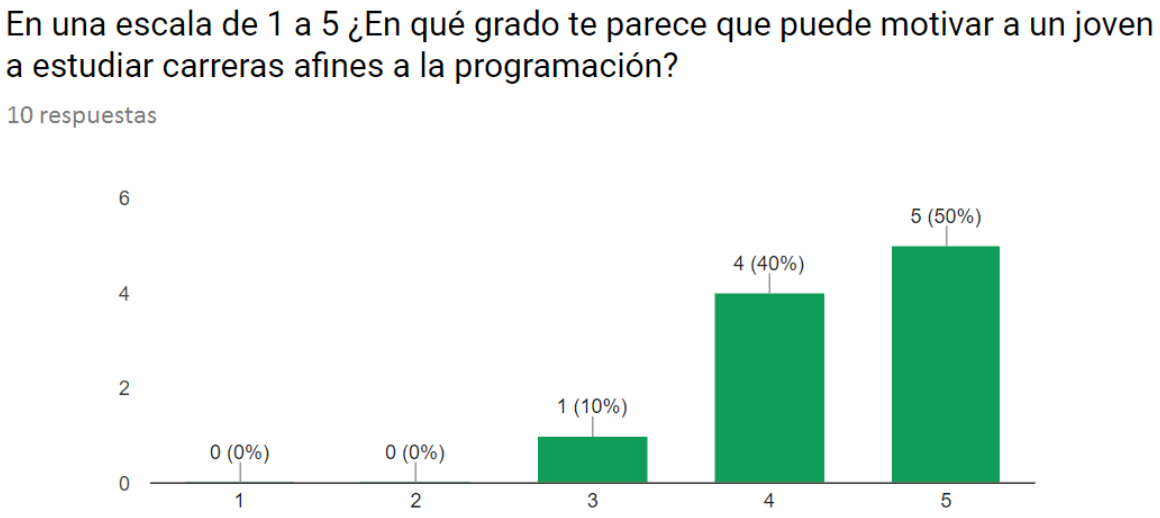

Figura 6.9: Grado en que puede motivar a un joven según los participantes.

Continuando con la encuesta, se les preguntó a los participantes si recomendarían el juego a otra persona. En este caso todos los encuestados respondieron que sí (ver Fig. 6.10).

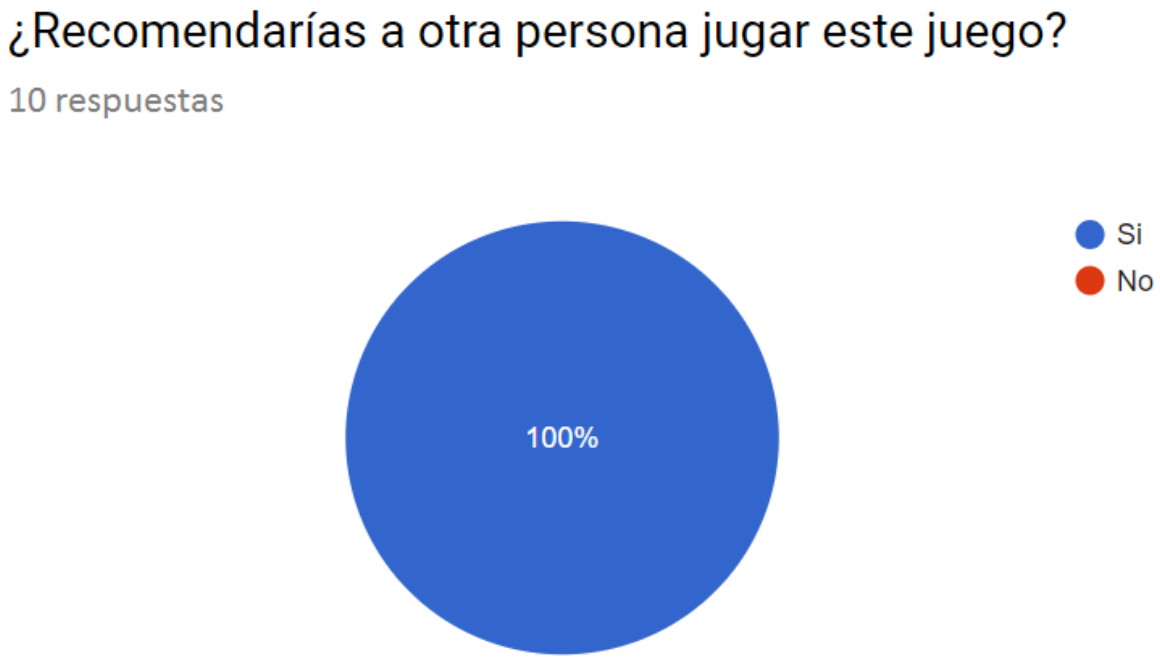

Figura 6.10: Cantidad de participantes que recomendarían el juego a otras personas.

En esta siguiente pregunta, se les pidió a los participantes que indiquen que conceptos,

de entre los que propone trabajar el juego, sentían que habían aprendido o reforzado. Los conceptos eran:

- Construir programas.
- La decisión (estructura de control *si*).
- La repetición (estructura de control *repetir*).
- El mientras (estructura de control *mientras*).
- Hardware y Software.
- Lenguajes de programación.

Como se puede observar en la Fig. 6.11, los encuestados sintieron que los conceptos que habían podido trabajar en mayor medida eran aquellos que se transmitieron de forma más práctica que teórica. De esto último se puede extraer que los conceptos de Hardware y Software y Lenguajes de programación, que son aquellos transmitidos mediante las misiones secundarias, son un aspecto del juego que debe ser mejorado, quizás agregando un componente más dinámico e interactivo para captar el interés de los jugadores.

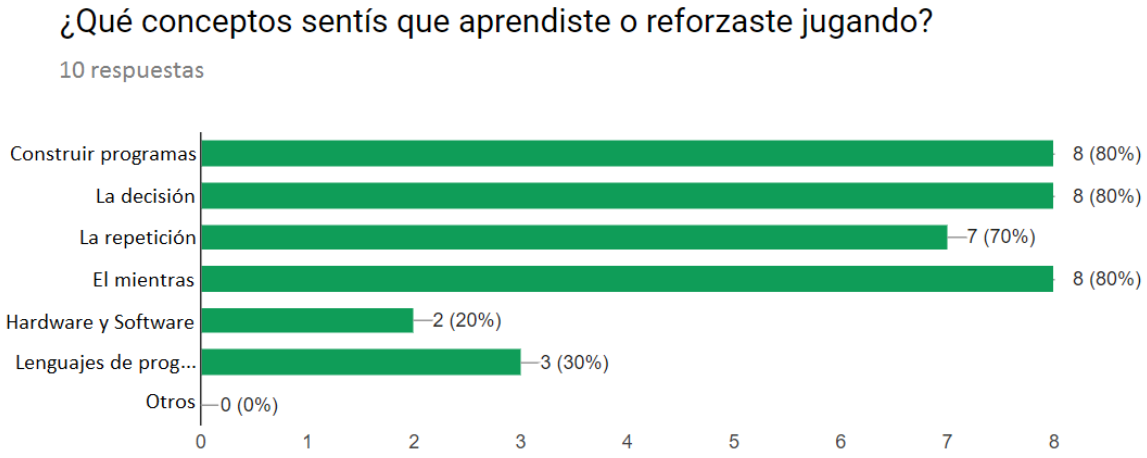

Figura 6.11: Conceptos que los participantes sienten que aprendieron con el juego.

Por último en esta primera parte de la encuesta, se incluyó un espacio para que el encuestado escriba, si así lo deseaba, un texto contando sobre los puntos fuertes y débiles que notó mientras jugaba el juego.

De los 10 participantes, 4 de ellos escribieron comentarios y de ellos se pudo extraer que, dado que los encuestados eran en su mayoría alumnos de carreras informáticas, estaban interesados en que el juego abordara temas más complejos y avanzados. También remarcaron que la experiencia les resultó amena e interesante, incluyendo algunos comentarios referidos a temas puntuales sobre los que se puede mejorar, los cuales se tendrán en cuenta para futuras versiones del juego.

6.4.2. Segunda parte de la Encuesta: instrumento IMI (*Intrinsic Motivation Inventory*)

Para la segunda parte de la encuesta se optó por seguir el instrumento IMI. Para describir este instrumento se debe primero definir qué es la motivación intrínseca. Se puede definir a la motivación como aquello que mueve a uno mismo o a otros a actuar. Las personas son a menudo movilizadas por factores externos tales como un sistema de recompensas, calificaciones, evaluaciones, o las opiniones que temen que otros tendrán sobre ellos. Pero, con la misma frecuencia, la gente es motivada desde el interior, por sus intereses, curiosidad y valores propios. Estas últimas son las llamadas motivaciones intrínsecas (Deci & Ryan, 1985, 2000). IMI es un instrumento de medición multidimensional utilizado para evaluar la experiencia subjetiva de los participantes en una actividad realizada en experimentos de laboratorio. Ha sido utilizado en variedad de experimentos relacionados con la medición de la motivación intrínseca (Ryan, 1982; Ryan, Mims & Koestner, 1983; Plant & Ryan, 1985; Ryan, Connell & Plant, 1990; Ryan, Koestner & Deci, 1991; Deci, Eghrari, Patrick & Leone, 1994).

El instrumento IMI se puede utilizar para evaluar el interés/disfrute, competencia percibida, esfuerzo, sentimiento de valor/utilidad, sentimiento de presión/tensión, y el sentimiento de elección percibido mientras se realiza la actividad, así se conforman seis subescalas de puntajes.

La subescala de interés/disfrute es considerada aquella que mide la autoevaluación de la

motivación intrínseca; por lo tanto, aunque el cuestionario en general es llamado Inventario de Motivación Intrínseca, es solamente esta subescala la que evalúa la motivación intrínseca en sí. Como resultado, la subescala de interés/disfrute a menudo tiene más ítems que las otras subescalas. Los conceptos de sentimiento de elección percibido y competencia percibida son teóricamente buenos para predecir tanto la autoevaluación como también mediciones de comportamiento sobre la motivación intrínseca, y presión/tensión es teóricamente bueno para predecir la motivación intrínseca negativa.

El esfuerzo es una variable separada que es relevante para algunas de las preguntas sobre motivación. La subescala de valor/utilidad es utilizada en estudios de internalización (Deci, 1994), siendo la idea que las personas internalizan y se autorregulan con respecto a actividades que experimentan como útiles y valiosas para ellos.

El cuestionario IMI consiste de una variada cantidad de ítems de estas subescalas, los cuales han demostrado ser analiticamente coherentes y estables a través de una variedad de tareas, condiciones y configuraciones. Existen, a su vez, versiones específicas de IMI que han sido usadas en estudios anteriores, cada una con diferente cantidad de ítems y subclases, y dirigidas a diferentes actividades. De entre estas versiones se seleccionó la versión estándar de 22 ítems con 4 subescalas: interés/disfrute, competencia percibida, sentimiento de elección percibido, y presión/tensión.

Los ítems del cuestionario IMI se presentan en forma de afirmaciones, y las pertenecientes a la versión seleccionada son:

1. Mientras realizaba la actividad pensaba cómo lo disfrutaba.
2. No sentí ninguna presión cuando hice la actividad.
3. Sentí que yo elegí hacer la actividad.
4. Creo que soy bastante bueno en la actividad.
5. La actividad me pareció muy interesante.
6. Sentí tensión al hacer la actividad.
7. Si me comparo con otros compañeros, creo que hice muy bien la actividad.

8. Me divirtió hacer la actividad.
9. Me sentí distendido al hacer la actividad.
10. Disfrute mucho hacer la actividad.
11. Tuve que hacer la actividad.
12. Estoy satisfecho en mi desempeño al hacer la actividad.
13. Me sentí ansioso al hacer la actividad.
14. Considere la actividad aburrida.
15. Sentí que podía hacer lo que quería al hacer la actividad.
16. Me sentí muy hábil al hacer la actividad.
17. Considere la actividad muy interesante.
18. Me sentí presionado al hacer la actividad.
19. Sentí que tuve que hacer la actividad.
20. Describiría la actividad como muy entretenida.
21. Hice la actividad porque no tuve opción.
22. Después de realizar la actividad durante un rato, me sentí muy competente.

Estas afirmaciones se responden utilizando una escala de 1 a 7. Siendo 1 la respuesta que representa que la afirmación correspondiente es “Nada cierta”, 4 representa “Algo cierta” y 7 representa “Muy cierta”.

La manera de evaluar los resultados de esta encuesta se describe a continuación: Se inicia puntuando a la inversa las preguntas número: 2, 9, 11, 14, 19, 21. En otras palabras, se resta 8 al puntaje de la pregunta, y se utiliza el resultado como el puntaje correspondiente. De esta manera, un puntaje más alto indica más del concepto descrito en la subescala nombrada. Así, un puntaje más alto en presión/tensión significa que la persona se sintió más presionada y tensionada; un puntaje más alto en competencia percibida significa que la persona se siente más competente realizando la actividad, y así sucesivamente. Después, se calcula el puntaje de cada subescala obteniendo el promedio de los puntajes de las preguntas que componen a cada subescala.

Las preguntas que corresponden a cada subescala son como sigue (los ítems marcados

con una “i” son aquellos que se deben calcular a la inversa):

- Interés/Disfrute: 1, 5, 8, 10, 14(i), 17, 20.
- Competencia percibida: 4, 7, 12, 16, 22.
- Sentimiento de elección percibido: 3, 11(i), 15, 19(i), 21(i).
- Presión/Tensión: 2(i), 6, 9(i), 13, 18.

Resultados de la Parte 2 – Cuestionario IMI

En la siguiente tabla 6.2 se presentan los resultados de la utilización del instrumento IMI seleccionado. Los participantes se presentan como las filas, etiquetadas con letras de la A a la J. Y las columnas se corresponden a las afirmaciones del instrumento antes nombradas, con aquellas afirmaciones que se deben calcular a la inversa resaltada en rojo.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
A	7	4	7	6	7	3	1	7	7	7	2	7	7	1	7	7	7	1	6	7	1	6
B	4	7	7	7	7	1	7	6	6	6	3	6	3	1	4	6	6	1	1	5	1	3
C	7	6	7	7	7	1	7	7	7	7	7	7	5	1	5	7	7	1	4	7	1	7
D	4	7	7	4	7	1	4	7	7	7	1	7	5	1	3	4	7	1	1	6	1	3
E	3	7	7	3	7	6	3	7	7	7	7	6	6	1	4	4	7	2	1	7	1	1
F	6	7	7	7	7	1	4	7	7	7	7	7	1	1	7	7	7	1	1	7	1	7
G	5	7	7	5	7	1	4	7	1	7	1	7	7	1	7	7	7	1	1	7	1	1
H	4	7	7	3	7	3	3	7	7	7	1	4	6	1	6	3	7	2	2	7	1	3
I	4	6	4	6	6	6	4	6	6	4	5	6	4	1	3	3	5	4	4	4	2	5
J	4	3	7	4	7	5	1	7	1	7	7	5	7	1	7	7	7	1	1	7	3	4

Tabla 6.2: Procesamiento del cuestionario IMI para los 10 participantes.

Los totales para cada pregunta se presentan en la Tabla 6.3, donde las columnas resaltadas en rojo son el resultado de restarle a 80 el total de la columna de la tabla anterior.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
R	48	19	67	47	67	28	38	68	24	66	39	62	51	70	53	55	67	15	58	64	67	40

Tabla 6.3: Totales del cuestionario IMI.

Y así, dividiendo el total de cada subescala por la cantidad de preguntas que la componen se obtiene el promedio de la subescala. Este promedio dividido por la cantidad de participantes muestra el resultado para cada subescala:

- Interés/Disfrute:
  - Total: 48 + 67 + 68 + 66 + 70 + 67 + 64 = 450
  - Promedio de la subescala: 450 / 7 = 64,3
  - Resultado: 64,3 / 10 = **6,43**
- Competencia percibida:
  - Total: 47 + 38 + 62 + 55 + 40 = 242
  - Promedio de la subescala: 242 / 5 = 48,4
  - Resultado: 48,4 / 10 = **4,84**
- Sentimiento de elección percibido:
  - Total: 67 + 41 + 53 + 58 + 67 = 286
  - Promedio de la subescala: 286 / 5 = 57,2
  - Resultado: 57,2 / 10 = **5,72**
- Presión/Tensión:
  - Total: 19 + 28 + 24 + 51 + 15 = 137
  - Promedio de la subescala: 137 / 5 = 27,4
  - Resultado: 27,4 / 10 = **2,74**

Como se explicó anteriormente, mientras más cerca del tope de 7 estén los puntajes, más se percibió el sentimiento nombrado. De esta manera, con los datos obtenidos se puede observar que hubo un nivel de interés/disfrute muy alto, la competencia percibida por los participantes fue buena y su sentimiento de elección durante la prueba fue alto. Se puede deducir entonces que el juego tiene un nivel de motivación satisfactorio.

El índice de presión/tensión, que representa el factor negativo dentro de las subescalas, se mantuvo en niveles bajos aunque casi alcanzando la mitad de la escala. Se puede observar que en especial la pregunta número 13 obtuvo niveles altos, mostrando que los participantes sintieron un nivel de ansiedad elevado durante la prueba, lo cual es interesante de analizar. Contar con filmaciones de la realización de la prueba resulta de gran utilidad en este caso, ya que se pueden observar si existieron factores externos al juego que pudieron contribuir a incrementar la tensión que experimentaron los participantes. En este caso, los alumnos venían de participar de una consulta previa, y es posible que arrastraran cierto cansancio sobre el final del día, dado que la prueba se realizó alrededor de las 18hs.

6.4.3. Entrevistas informales y Filmaciones

El diálogo con los participantes en forma posterior, mostró también interés de los participantes en el juego, su dinámica y en que se avanzara en la inclusión de temas avanzados. Al mismo tiempo, las filmaciones permitieron registrar tiempos de realización de las tareas, comprobando que la sesión duró alrededor de una hora incluyendo la introducción del juego dada por los tesistas y las entrevistas finales, siendo el tiempo promedio de realización de todos los escenarios fue de 30 minutos aproximadamente.

6.5. Conclusiones del capítulo

En este capítulo se presentó el diseño de las sesiones de prueba realizadas durante el proceso de desarrollo y al finalizar el mismo. Al mismo tiempo, se presentaron los

resultados a partir de la recogida de datos realizada con los diferentes instrumentos. Se puede afirmar que la experiencia de la prueba fue positiva, ya que los resultados arrojaron que los participantes mostraron un índice de motivación alto y encontraron el juego útil, y educativo. Los participantes remarcaron haber profundizado en la comprensión de las estructuras de control de decisión, repetición e iteración. Al mismo tiempo, todos los participantes expresaron que recomendarían el juego a sus compañeros y les gustaría continuar jugando. Se interesaron en conocer cómo continuaba la historia lo que demuestra una motivación por la historia presentada. Por otra parte, el hecho que el índice de presión/tensión llegara a la mitad de la escala, deja la puerta abierta para seguir profundizando y analizar la configuración de la sesión o qué aspectos hicieron que los participantes sintieran esto. Se continuará trabajando en sesiones de prueba y en la difusión del juego para avanzar en el análisis de los resultados y obtener resultados más concluyentes. También restan hacer pruebas del generador de escenarios con docentes e interesados en general. Se espera continuar entonces con la evolución de AstroCódigo.

CAPÍTULO 7

Aportes, conclusiones y trabajos futuros

7.1. Introducción

En este capítulo se resumen los aportes de esta tesina, así como también las conclusiones a las que se han llegado en las distintas etapas transcurridas a lo largo de su desarrollo. Además, se proponen aspectos a mejorar y trabajos futuros que se vinculan con las líneas de acción desarrolladas.

7.2. Aportes

Este trabajo presenta como principal aporte el desarrollo de AstroCódigo, un juego serio para introducir a estudiantes en conceptos de programación, y que se considera un aporte tanto para la comunidad educativa, así como para los seguidores y desarrolladores de videojuegos. Al mismo tiempo, se pone a disposición una herramienta web que constituye un complemento para docentes, y que permite personalizar escenarios de programación para incluir en AstroCódigo. Finalmente, se expone una serie de análisis y conclusiones a las que se arriba luego de haber puesto en juego a AstroCódigo con alumnos y la revisión teórica realizada que pueden ayudar a quienes investigan estas temáticas.

7.3. Conclusiones

El uso de juegos serios orientados a objetivos educativos ha demostrado tener beneficios en numerosas experiencias en las cuales se aplicaron. En la mayoría de los estudios revisados se da cuenta de un aumento en el nivel de motivación e interés sobre la temática presentada, superior en aquellos individuos que utilizaron juegos serios a los que no. En este trabajo se investigaron particularmente aquellos orientados al aprendizaje en conceptos de programación, cuyos casos de éxito sirvieron tanto de inspiración como de orientación, y su análisis permitió considerar aspectos deseados como requerimientos para el diseño de AstroCódigo.

Para el diseño del juego presentado en este trabajo, se decidió que los objetivos del mismo estarían orientados al aprendizaje de conceptos básicos de programación en jóvenes de edad escolar. Para lograr esto se sopesaron muchas ideas, de las cuales se tuvieron que elegir algunas y descartar otras, sin embargo, fue claro desde un principio que el juego intentaría lograr un balance justo entre aprendizaje y entretenimiento, siendo éste último una característica usualmente desbalanceada en muchos juegos serios. Este proceso llevó aproximadamente un mes en el que se llegó a una idea general del juego, para luego ir evolucionando a medida que se avanzaba en la implementación del mismo.

El desarrollo del juego tomó más tiempo del esperado, abarcando aproximadamente ocho meses de trabajo, acorde a la envergadura que fue tomando la propuesta. Durante este proceso se enfrentaron numerosos desafíos y dificultades que ameritaron días enteros de trabajo en ciertas funcionalidades y cambios de rumbo en cuanto a características del juego. Para lograr el resultado conseguido, fue necesario el uso de diversas herramientas y lenguajes de programación, lo cual implicó un proceso de aprendizaje muy interesante que abarcó desde el uso de un *framework* para el desarrollo web como *Laravel*, el uso de softwares de diseño 3D y 2D como lo son *Blender* y *Piskel* respectivamente, hasta el uso de un motor de videojuegos como *Unity*. Las herramientas de desarrollo y modelado se consideran un acierto y han permitido plasmar y acompañar las tareas propuestas.

A partir de las sesiones de prueba realizadas con jóvenes de diversas edades y con distintos niveles de conocimientos previos en programación, se vislumbra que el juego logra ayudar en el aprendizaje, la profundización, y puesta en juego de los conceptos y habilidades propuestas en los objetivos, al mismo tiempo que genera una experiencia motivante.

El trabajo realizado en esta tesina tiene como finalidad, además de permitir la culminación de las carreras de grado de ambos autores, brindar un aporte a la sociedad, poniendo a disposición de la comunidad una herramienta para aquellos que deseen introducirse a la programación o para quien quiera enseñar estas temáticas.

La difusión de esta herramienta será esencial para su real aprovechamiento. Es por ello, que se ha realizado la presentación de un trabajo a un congreso (Congreso EduLearn, España) y se presentará como demo educativa en el Congreso de Tecnología en Educación y Educación en Tecnología en nuestro país.

7.4. Trabajos futuros

Si bien todas las funcionalidades diseñadas para el juego fueron implementadas, hay espacio para mejoras y adiciones al mismo que pueden ser abordadas como trabajos futuros. Entre las futuras líneas de acción se pueden destacar las siguientes:

Respecto de mejoras en los desarrollos realizados:

- Por cuestiones de alcance y de tiempos, los conceptos relacionados a variables y módulos son introducidos solo de manera teórica en el juego. Sería interesante para un trabajo futuro integrar estos conceptos a las mecánicas de juego de los escenarios de programación, brindando la posibilidad, quizás, de utilizar variables en el programa y crear módulos que permitan reutilización de código. Agregar esta funcionalidad implicaría un tiempo considerable de trabajo. Por ejemplo, se debería pensar y diseñar

nuevos elementos como robots o casilleros, que ameriten el uso de estas nuevas mecánicas.

- Otro punto que podría ser abordado en un trabajo futuro es el diseño del sitio web del juego, que si bien se intentó darle una estética conforme al juego, no se invirtió el suficiente tiempo su diseño. Un mejor diseño en el sitio web podría significar una mayor atracción hacia el juego por parte de los potenciales jugadores.

- Por último, una pequeña mejora más inmediata sería analizar y perfeccionar el algoritmo que permite la detección de patrones de instrucciones, y que permite visualizar el programa óptimo para resolver un escenario de programación en la herramienta web de creación de escenarios personalizados, ya que existen casos específicos en los cuales el algoritmo arroja resultados inexactos provocando que las estructuras de control no se aniden correctamente.

Respecto de las investigaciones realizadas:

- Resulta necesario profundizar en los resultados alcanzados con el trabajo con Astro-Código, a partir de extender la muestra de participantes. En este sentido se planifica realizar sesiones con alumnos de nivel medio y con ingresantes a la Facultad de Informática.

- También es necesario realizar sesiones de prueba con el generador de escenarios, a partir del trabajo con docentes afines a la enseñanza de la programación, de manera tal de evaluar sus potencialidades y sus barreras. Este aspecto no se alcanzó a realizar en el trabajo. Sólo se hicieron pruebas iniciales con los directores.

- Finalmente, se cree que la investigación teórica puede profundizarse a partir de la lectura de trabajos y experiencias que puedan haberse realizado en forma posterior a la indagación realizada aquí o que no se hayan considerado en esta tesina por la amplitud del tema.

Referencias

- Abt C.C. (1970). *Serious Games*. Prensa Universitaria de América. Estados Unidos.
- Aguil Mallea D., Depetris B. O., Feierherd G. E., Pendenti H., Prisching G., Tejero G. (2015). *La enseñanza y el aprendizaje de la programación y la programación concurrente con DaVinci Concurrente*. X Congreso sobre Tecnología en Educación & Educación en Tecnología (TE & ET). Corrientes, Argentina.
- Aho A.V., Hopcroft J.E., Ullman J.D. (1988). *Estructuras de datos y algoritmos*. Pearson Educación.
- Ander Egg E. (1993). *La planificación educativa. Conceptos, métodos, estrategias y técnicas para educadores*. Universidad Nacional de Mar del Plata. Buenos Aires, Argentina.
- Anderson L. W. (2005). *Objectives, evaluation, and the improvement of education*. Universidad de Carolina del Sur. Estados Unidos.
- Artola V. (2013). *Interacción tangible en aplicaciones educativas. Diseño e implementación de un prototipo basado en este paradigma de interacción orientado al aprendizaje colaborativo*. Facultad de Informática, Universidad Nacional de La Plata. Argentina.
- Artola V., Gorga G., Pesado P., Sanz C. (2014). *Diseño de un juego basado en interacción tangible para la enseñanza de programación*. Instituto de Investigación en Informática LIDI, Facultad de Informática, Universidad Nacional de La Plata. Argentina.
- Bacon L., Kazimoglu C., Kiernan M., Mackinnon L. (2012). *A serious game for developing computational thinking and learning introductory computer programming*. Universidad de Greenwich. Londres, Reino Unido.

- Bloom B. S., Engelhart M. D., Furst E. J., Hill W. H., Krathwohl D. R. (1913). *Taxonomía de los objetivos de la educación: La clasificación de las metas educacionales*. Universidad de Chicago. Estados Unidos.

- Brown Bartneche M., Fava L., Gómez S., Miyuki Kimura I., Queiruga C. (2014). *El juego como estrategia didáctica para cercar la programación a la escuela secundaria*. Laboratorio de Investigación en Nuevas Tecnologías Informáticas, Facultad de Informática, Universidad Nacional de La Plata. Argentina.

- Cámara de Empresas de Software & Servicio Informáticos de La República Argentina (2014). *Reporte anual del sector de software y servicios informáticos de la República Argentina*. Consultado en febrero de 2016. Recuperado de: http://www.cessi.org.ar/descarga-institucionales1838/documento28ecd0d438429d63951a021005fbcd7cd

- Champredonde R., De Giusti A., (1997). *Design and Implementation of the Visual DaVinci Language*. III Congreso Argentino de Ciencias de la Computación. Buenos Aires, Argentina.

- Chen S., Michael D. (2006). *Serious Games: Games that educate, train, and inform*. Thomson Course Technology. Boston, Estados Unidos.

- Chichizola, F., De Giusti, A. E., De Giusti, L. C., Leibovich, F., Naiouf, M., Rodriguez, S., Sanchez, M. (2014). *Herramienta interactiva para la enseñanza temprana de concurrencia y paralelismo: un caso de estudio*. XX Congreso Argentino de Ciencias de la Computación. Buenos Aires, Argentina.

- Darhmaoui H., Elachqar A., Kaddari F., Lahmine S., Ouahbi I. (2014). *Learning basic programming concepts by creating games with Scratch programming environment*. Laboratorio de Didactica, Innovación, Pedagogía y Curricula, Facultad de Ciencia Dhar El Mahraz, Universidad Sidi Mohamed Ben Abdellah. Marruecos.

- Deci, E. L., Eghrari, H., Patrick, B. C., Leone, D. (1994). *Facilitating internalization: The selfdetermination theory perspective*. Journal of Personality.

- Deci, E. L., Ryan, R. M. (1985). *Intrinsic motivation and self-determination in human behavior*. Nueva York, Estados Unidos.
- Deci, E. L., Ryan, R. M. (2000). *The "what" and "why" of goal pursuits: Human needs and the self-determination of behavior*. Psychological Inquiry.
- Deci, E. L., Ryan, R. M. (2000). *Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being*. American Psychologist.
- Díaz J., Queiruga C. & Fava L. (2015). *Juegos Serios y Educación*. XVII Workshop de Investigadores en Ciencias de la Computación. Salta. Argentina: RedUNCI.
- Dörner R., Effelsberg W., Göbel S., Wiemeyer J. (2016). *Serious Games. Foundations, concepts and practice*. Springer. Alemania.
- Gomes F., Pereira Lopes M. & Vaz de Carvalho C. (2013). *Serious Games for Lean Manufacturing: The 5S Game*. Revista Iberoamericana de tecnologías del aprendizaje, Vol 8.
- Gorga G., Salazar Mesía N., Sanz C. (2015). EPRA: *Herramienta para la enseñanza de conceptos básicos de programación utilizando realidad aumentada*. Instituto de Investigación en Informática LIDI, Facultad de Informática, Universidad Nacional de La Plata. Argentina.
- Greuenbaum P. (2014). *Undergraduates teach game programming using Scratch*. Sociedad de computación IEEE. Estados Unidos.
- Herz, J. (1997). *Joystick Nation*. Londres
- Huizinga J. (1938). *Homo ludens*. Holanda.
- Jessel J.P., Mauratet M., Torguet P., Viallet F. (2011). *Experimental feedback on Prog&Play: A serious game for programming practice*. Universidad Paul Sabatier. Toulouse, Francia.
- Johnson, S. (1997). *Interface Culture*. Basic Books, Nueva York, Estados Unidos.

- Kereki I.F. (2008). *Scratch: Applications in Computer Science 1*. Universidad ORT de Uruguay. Uruguay.
- Krathwohl D. R. (2002). *A revision of Bloom's taxonomy: An overview. Theory into practice, volume 41, number 4*. Universidad del Estado de Ohio. Estados Unidos.
- Leon-Sigg M., Ramírez-Rosales S., Vazquez-Reyes S., Villa-Cisneros J. L (2016). *A serious game to promote object oriented programming and software engineering basic concepts learning*. Universidad Autónoma de Zacatecas. México.
- López M., Del Olmo P., Reyes C. & Fernández E. (2013). *La programación lúdica como estrategia de articulación entre niveles*. VIII Congreso de Tecnología en Educación y Educación en Tecnología. Santiago del Estero. Argentina: RedUNCI
- Malone, T. (1981). *Towards a theory of intrinsically motivating instruction*. Cognitive Science.
- Marzano, R. J. y Kendall, J.S. (2007). *The new taxonomy of educational objectives*. California, Estados Unidos.
- McAuley, E., Duncan, T., Tammen, V. V. (1987). *Psychometric properties of the Intrinsic Motivation Inventory in a competitive sport setting: A confirmatory factor analysis*. Research Quarterly for Exercise and Sport.
- Ministerio de Ciencia, Tecnología e Innovación Productiva. (2014). *Análisis de Diagnóstico Tecnológico Sectorial (ATS)*. Consultado en febrero de 2016. Recuperado de: http://www.mincyt.gob.ar/informes/analisis-de-diagnostico-tecnologico-sectorial-ats-11641
- Plant, R. W., Ryan, R. M. (1985). *Intrinsic motivation and the effects of self-consciousness, self-awareness, and ego-involvement: An investigation of internally-controlling styles*. Journal of Personality.
- Prensky, M. (2001). *Digital Game Based Learning*. McGraw-Hill, Nueva York, Estados Unidos.

- Provenzo, E. (1991). *Video Kids Making Sense of Nintendo*, Harvard University Press.
- Queiruga C., Fava L., Gómez S., Miyuki Kimura I. & Brown Bartneche M. (2014). *El juego como estrategia didáctica para acercar la programación a la escuela secundaria*. XVI Workshop de Investigadores en Ciencias de la Computación. Tierra Del Fuego, Argentina: RedUNCI.
- Quinn, C. (1996). *Designing an instructional game: reflections on Quest for Independence*. Education and Information Technologies.
- Raths L. E., Wassermann S. (1967). *Cómo enseñar a pensar: Teoría y aplicación*. Columbus, Ohio, Estados Unidos.
- Rugelj J., (2015). *Serious computer games in computer science education*. Universidad de Liubliana. Liubliana, Eslovenia.
- Ryan, R. M. (1982). *Control and information in the intrapersonal sphere: An extension of cognitive evaluation theory*. Journal of Personality and Social Psychology.
- Ryan, R. M., Connell, J. P., Plant, R. W. (1990). *Emotions in non-directed text learning*. Learning and Individual Differences.
- Ryan, R. M., Koestner, R., Deci, E. L. (1991). *Varied forms of persistence: When freechoice behavior is not intrinsically motivated*. Motivation and Emotion.
- Ryan, R. M., Mims, V., Koestner, R. (1983). *Relation of reward contingency and interpersonal context to intrinsic motivation: A review and test using cognitive evaluation theory*. Journal of Personality and Social Psychology.
- Seel N. M. (2012). *Encyclopedia of the sciences of learning*. Springer. Estados Unidos.
- Waraich, A. (2004). *Using Narrative as a Motivating Device to Teach Binary Arithmetic and Logic Gates*. Universidad Metropolitana de Manchester.

Anexo: Encuesta

1	Marca temporal	En una escala de 1 a 5 ¿Qué tan familiarizado estabas con los conceptos de programación presentados en el juego antes de jugarlo?	¿Cuántos escenarios resolviste?	¿Qué cantidad de misiones resolviste?	En una escala de 1 a 5 ¿Cómo clasificarías la dificultad del juego?	En una escala de 1 a 5 ¿Cómo ponderarías la utilidad de los video-tutoriales en el juego?	En una escala de 1 a 5 ¿Qué tan intuitiva te pareció la interfaz del juego?	En una escala de 1 a 5 ¿Cómo clasificarías la utilidad del juego a nivel educativo?
2	18/11/2016 18:10:51	5	6	2	2	5	4	5
3	18/11/2016 18:11:11	5	6	2	3	4	4	5
4	18/11/2016 18:14:17	5	4	2	2	3	3	5
5	18/11/2016 18:14:37	4	6	2	2	4	5	5
6	18/11/2016 18:17:29	5	6	2	3	5	5	5
7	18/11/2016 18:18:54	5	6	2	3	4	4	5
8	18/11/2016 18:21:06	5	6	1	3	3	5	5
9	18/11/2016 18:23:03	5	5	1	2	4	5	5
10	18/11/2016 18:30:56	1	6	2	3	2	3	4
11	18/11/2016 18:31:01	1	6	2	3	3	3	5

1	En una escala de 1 a 5 ¿En qué grado sentiste que no sabías cómo proseguir con el juego?	En una escala de 1 a 5 ¿En qué grado te parece que puede motivar a un joven a estudiar carreras afines a la programación?	¿Recomendarías a otra persona jugar este juego?	Si querés, contanos sobre los puntos fuertes y débiles que encontraste en el juego.	¿Qué conceptos sentís que aprendiste o reforzaste jugando?	Dejanos tu nombre y apellido, edad y carrera a la que te anotaste
2	1	5	Si		Construir programas., La repetición (estructura de control REPETIR)., Hardware y Software., Lenguajes de programación.	Agustin Casas, 20, Lic. Informatica
3	1	3	Si	Esta muy buena la interfaz de programación, con las opciones, los tutoriales, el poder mover la camara, y me gustó la posibilidad de hablar con personajes cuando nos acercábamos. Estaría bueno armar algunos niveles más complicados donde se pueda usar por ejemplo un repetir adentro de otro, y demás.	La decisión (estructura de control SI).	Almandos, Julian - 18 - Licenciatura en Sistemas
4	1	5	Si	Me gustaria que el juego incorporara algunos otros conceptos de la programacion, creo que los mas basicos estan muy bien abordados.	Construir programas., La decisión (estructura de control SI)., La repetición (estructura de control REPETIR)., El mientras (estructura de control MIENTRAS).	Nahuel Mario Herpo, edad 18, Licenciatura en Informática
5	2	4	Si	Entretenido, interesante. No me quedo claro si a la hora de eliminar una instruccion del programa había que deslizarla hacia afuera o hacer otra cosa. Lo demás me pareció genial.	Construir programas., La decisión (estructura de control SI)., La repetición (estructura de control REPETIR)., El mientras (estructura de control MIENTRAS).	Lucas Sanz 18 Ingeniería Electrónica
6	2	4	Si		Construir programas., La decisión (estructura de control SI)., La repetición (estructura de control REPETIR)., El mientras (estructura de control MIENTRAS).	Agustin Ramirez. 17 años. Licenciatura en informatica
7	2	5	Si	Puntos fuertes: el uso de la programación en bloques que es muy util para la enseñanza de la programación. Además el drag & drop lo hace muy intuitivo. La idea de los robots especializados le dan una variabilidad interesante que puede ser muy explotada. Los guiños a cosas argentas pueden llegar a ser atrayentes. Milanesa!! EL PERRO!!!! Puntos débiles: en el modo historia la mision para conseguir el robot de laser esta medio escondida. Por ahi con un guiño a su ubicacion alcanzaría para verla mas fácil (tal vez poner el efecto de los puntos flotando mas alto?). Las misiones la idea es interesante pero no parecen muy atrayentes. Tal vez con una recompensa podrían sacarsele mas jugo. Aclaración: la pregunta de facilidad la conteste pensando como alguien que no sabe poco o nada del tema.	Construir programas., La decisión (estructura de control SI)., La repetición (estructura de control REPETIR)., El mientras (estructura de control MIENTRAS).	Federico Archuby, 27, Lic. en Sistemas.
8	2	4	Si		La repetición (estructura de control REPETIR)., El mientras (estructura de control MIENTRAS).	Roberto Bascuñán, 18 Años, Licenciatura en Sistemas
9	2	5	Si		Construir programas., La decisión (estructura de control SI)., La repetición (estructura de control REPETIR)., El mientras (estructura de control MIENTRAS)., Lenguajes de programación.	Tomas Sandler, 17 años, Licenciatura en Informatica
10	2	4	Si		Construir programas., La decisión (estructura de control SI)., El mientras (estructura de control MIENTRAS)., Hardware y Software.	Iara Buffarini, 20, prof educacion fisica
11	4	5	Si		Construir programas., La decisión (estructura de control SI)., El mientras (estructura de control MIENTRAS)., Lenguajes de programación.	Agustina Paladini. 19años. Diseño de Indumentario